AGILE & SCRUM

KNOW ABOUT THE PROCESS & FRAMEWORKS

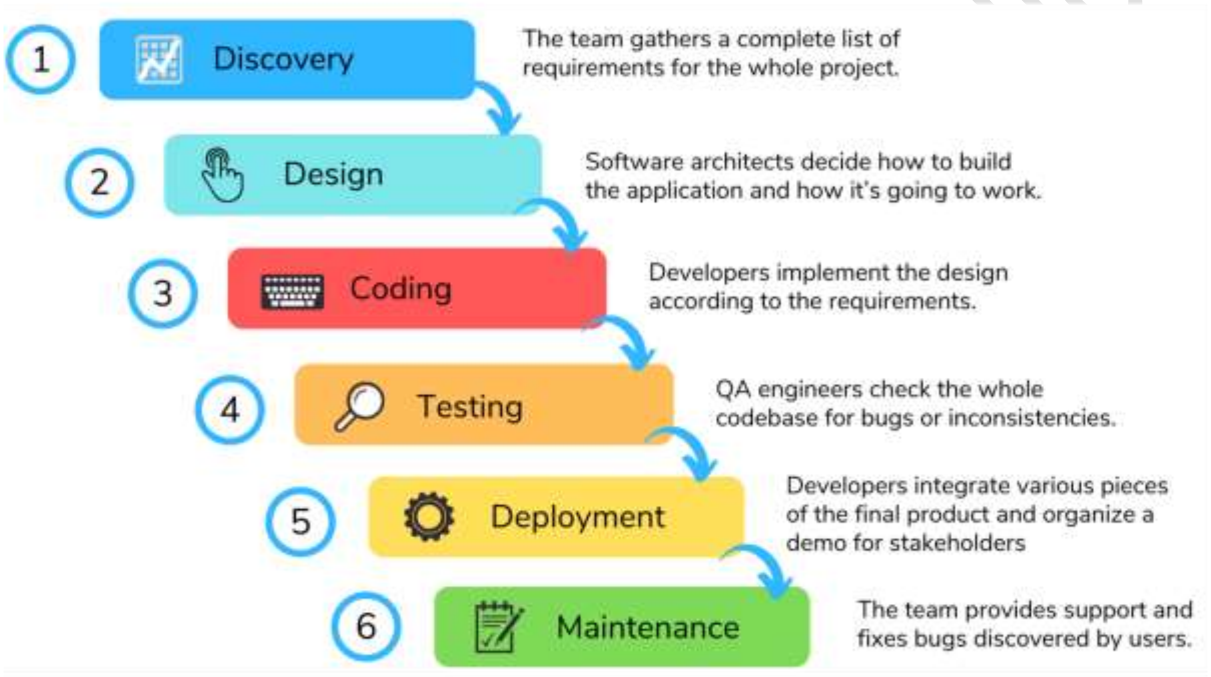
"With agile, we are uncovering better ways of developing software by doing it and helping others do it."

Agile way of thinking

Introduction

Agile, represents the adaptiveness and response to change, i.e., being flexible. In the early 2000s, many projects were failing or taking much too long to complete.

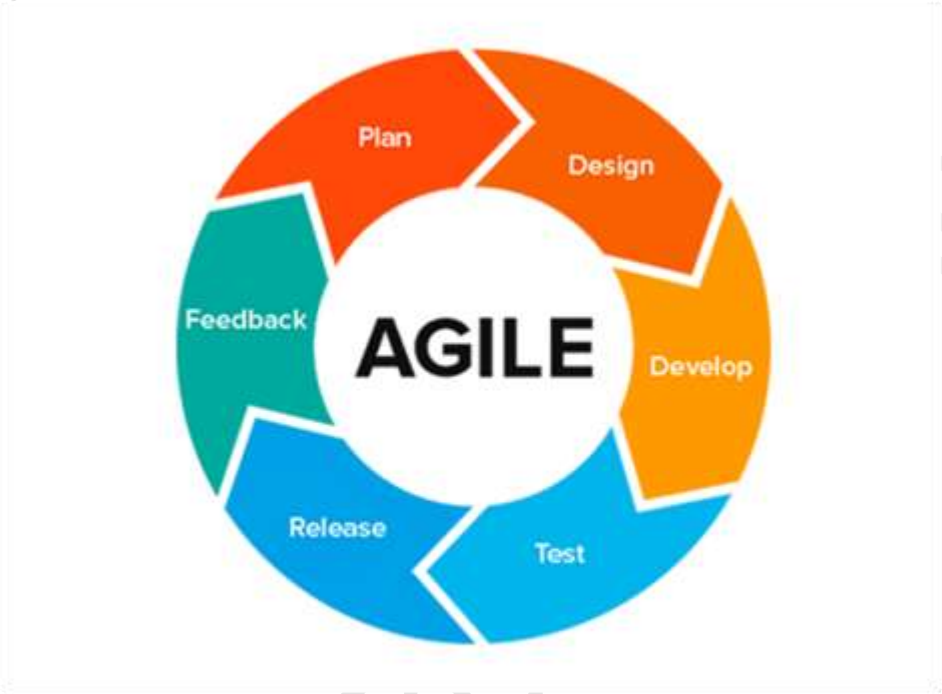
- Product-market fit problem: by the time software was developed, customer would have lost this precious time to check its fitment or customer requirements will change by then. What if after launch you realize that the product needs change according to the users?


1 Discovery
The team gathers a complete list of requirements for the whole project.
2 Design
Software architects decide how to build the application and how it's going to work.
3 Coding
Developers implement the design according to the requirements.
4 Testing
QA engineers check the whole codebase for bugs or inconsistencies.
5 Deployment
Developers integrate various pieces of the final product and organize a demo for stakeholders
6 Maintenance
The team provides support and fixes bugs discovered by users.

- Difficult to accept change: It was only after complete software was developed that you would actually know how users are using it. After putting in months, it was difficult to change quickly as per user needs & actual use cases.

This problem forced industry leaders to find a new, innovative approach.

- Iterative delivery, i.e., deliver value (logical pieces of working software) in iterations so as to resolve product-market fir problem. Users can actually start using the product with each increment, thereby confirming its usability/fitment for the purpose.



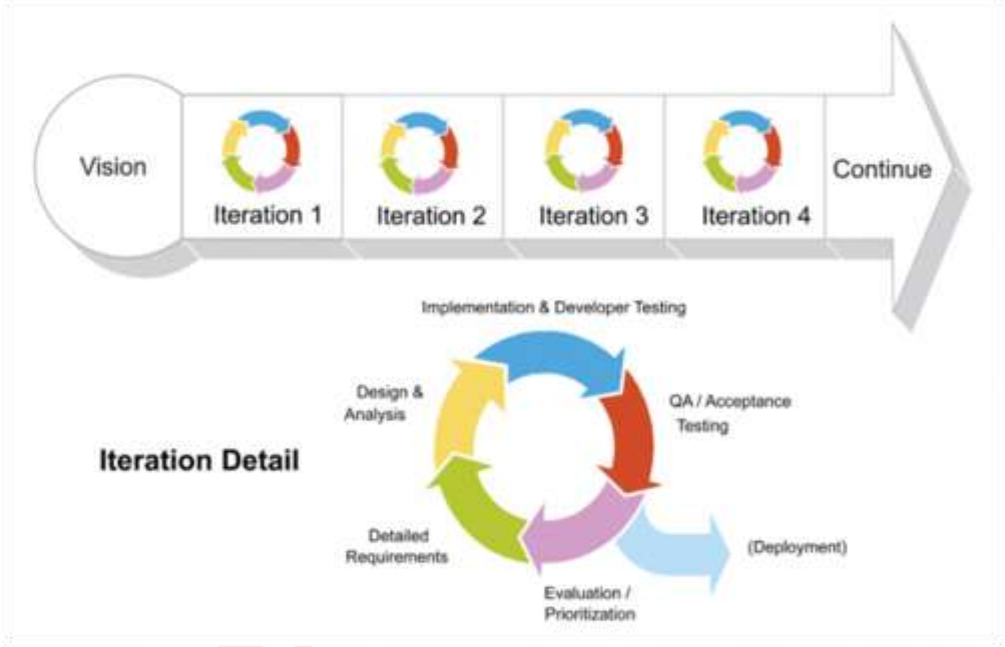
- Getting feedback from users quickly to confirm the usefulness of new software and continue to improve on it accordingly. I.e., being flexible to accommodate changes as per user feedback.

The entire project is finished in mini-projects (iterations) so the team has better control over the deliverables and is flexible to adapt to any changes. Develop > Demo > Adapt. Most of the projects today follow agile practices. Agile is a way of thinking, a mindset, the ability to shift your focus when and how the situation requires it.

What is agile

"A group of software development methodologies based on 'iterative' development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams".

- Group of methodologies: there's no one right way to implement agile. There are many different types of methodologies from which to choose – Kanban, Scrum, Extreme Programming (XP), Feature-driven development (FDD), Dynamic Systems Development Method (DSDM), Crystal, Lean.
- Iterative development: a '*repeated*' interval in which work items are fully developed and tested.



- Collaboration: connect the right people from different groups to work together on tasks, boosting cross-team collaboration and productivity.
- self-organizing cross-functional teams: As the name says, the team is able to craft their way of working (self-organizing) and is a mix of people who have access to all the skills necessary to effectively deliver value to customers (cross-functional).

'Agile' represents the adaptiveness and response to change, delivering business value through product 'increments' - at regular intervals. A mindset informed by the Agile Manifesto's values and principles.

Agile frameworks: Scrum, Crystal, Extreme Programming, Dynamic Systems Development Method (DSDM), Feature-Driven Development (FDD), Kanban.

Agile practices: Pair programming, Test-driven development, Stand-ups, Planning sessions, and Sprints.

Iteration and Increment are not the same thing!

Iteration: a 'repeated' interval (typically 1-4 weeks) in which work items are fully developed and tested. Projects are made up of as many iterations as needed.

Increment: Product increment, i.e., what gets produced at the end of a development period or timebox.

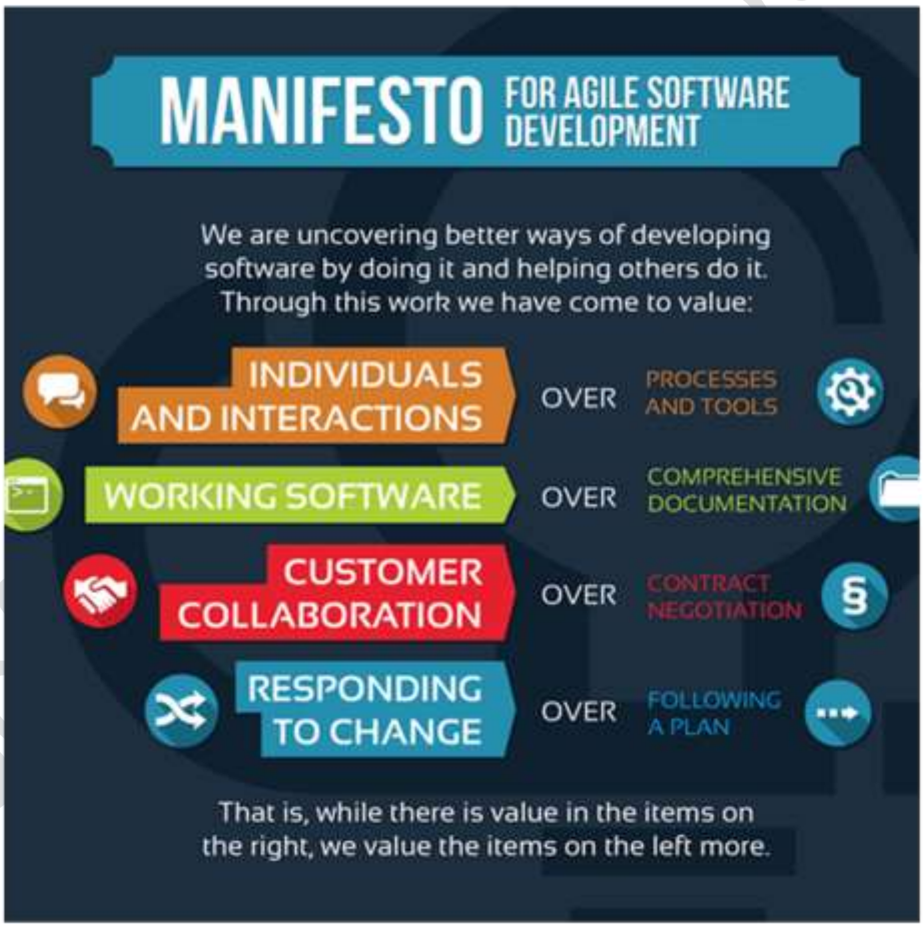
Agile Manifesto

The Agile Manifesto is a document that sets out the key values and principles behind the Agile philosophy and serves to help development teams work more efficiently and sustainably

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more".



MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS

WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION

CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION

RESPONDING TO CHANGE OVER FOLLOWING A PLAN

That is, while there is value in the items on the right, we value the items on the left more.

***Individuals and interactions** over processes and tools*

We should focus more on having competent people working together effectively. Processes and tools are not harmful but don't use them until they make things easy and effective.

***Working software** over comprehensive documentation*

Instead of wasting a lot of time creating lengthy documents and then taking feedback based on those documents, we should take the feedback based on working software.

***Customer collaboration** over contract negotiation*

When software is developed based on a pre-defined contract (without constant feedback), the result is often a product that does not completely satisfy the customers' needs. That is why Agile emphasizes to take constant feedback from the customer throughout the development process.

***Responding to change** over following a plan*

Start with a plan that gives us enough idea of the direction we want to move. Then as we proceed and seeing new events unfolding in front of us, we incorporate those realities into the ongoing work.

Agile principles 1-3

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

In the case of traditional methodologies, customers get to see the product only after which keeps the customers in dark till the end and also makes it problematic for the team members to introduce any changes in the product.

Instead engage customers throughout the development process by continuously delivering a working version of the product after every iteration. Show small increments and make changes as required.

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

Agile supports observing changing markets, customer needs, and competitive threats and changing course when necessary. Incorporating feedback or requirement changes is an ongoing process after every iteration.

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

Break product's development into smaller components that can be completed in a set timeframe (iteration or sprint) and deliver those components frequently.

Agile principles 4-6

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

Better decisions are made when the business and technical team are aligned.

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

Support, trust, and motivate the people involved – Motivated teams are more likely to deliver their best work than unhappy teams.

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

Enable face-to-face interactions – Communication is more successful when development teams are co-located.

Agile principles 7-9

Working software is the primary measure of progress.

The only factor to measure success with each iteration is the delivery of a working product. Ideally the product increment should satisfy the customer, or provide valuable feedback for next iteration.

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

Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.

Continuous attention to technical excellence and good design enhances agility.

The right skills and good design ensure the team can maintain the pace, constantly improve the product, and sustain change.

Agile principles 10-12

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

Develop "just enough" to get the job done for right now.

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

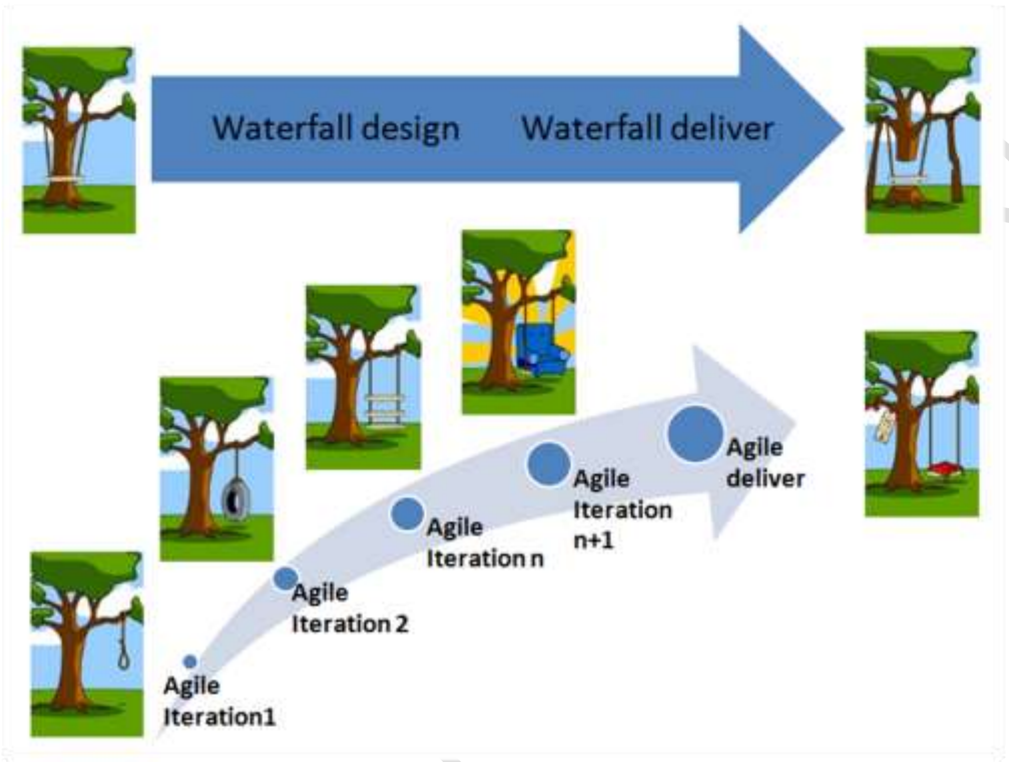
Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.

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

Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

Waterfall vs Agile

To put it simply Waterfall is essentially making a good plan and sticking to it, while Agile utilises a more flexible, iterative approach. Waterfall is more sequential and pre-defined, while Agile is more adaptable as a project progresses.



Waterfall (sequential-linear approach), is much more linear, focusing on ***up front planning*** with ***requirements fully defined*** before a project commences. Like its name suggests, ***work cascades***, much like a waterfall, through different project phases. Each ***phase needs to be completed*** before the next one can begin.

Agile (incremental-iterative approach), on the other hand, is a practice based on continuous ***iterations of development and testing*** where such activities can run concurrently. Agile projects are characterised by a series of tasks that are conceived, executed and ***adapted as the situation demands***, rather than a pre-planned process.

- Agile is an incremental and iterative approach; Waterfall is a linear and sequential approach.
- Agile separates a project into sprints; Waterfall divides a project into phases.
- Agile introduces a product mindset with a focus on customer satisfaction; Waterfall focuses on successful project delivery.
- Agile allows requirement changes at any time; Waterfall avoids scope changes once the project starts.
- Testing is performed concurrently with development in Agile; testing phase comes only after the build phase in a Waterfall project.

- Agile enables the project team to operate without a dedicated project manager; Waterfall requires a project manager who plays an essential role in every phase.

Benefits of agility

Stakeholder engagement: Stakeholders and team members have opportunities to observe and test throughout the project which allows for adjustments and changes to be made as things move forward. This greater "user focus" means that on delivery it's likely the outcome will be more in line with expectations - even if they have evolved along the way.

Focuses on Business Value: By allowing the client to determine the priority of features, the team understands what's most important to the client's business, and can deliver the features that provide the most business value.

Allows for Change: While the team needs to stay focused on delivering an agreed-to subset of the product's features during each iteration, there is an opportunity to constantly refine and reprioritize the overall product backlog. New or changed backlog items can be planned for the next iteration, providing the opportunity to introduce changes within a few weeks.

Collaboration: Agile encourages teamwork, collaboration, self-organisation and accountability. This helps with overall motivation and commitment to a project's outcomes and goals.

Customer satisfaction: Customer-focused approach, resulting in increased customer satisfaction.

Early and Predictable Delivery: By using time-boxed, fixed schedule iterations, new features are delivered quickly and frequently, with a high level of predictability. If time to market for a specific application is a greater concern than releasing a full feature set at initial launch, Agile can more quickly produce a basic version of working software which can be built upon in successive iterations.

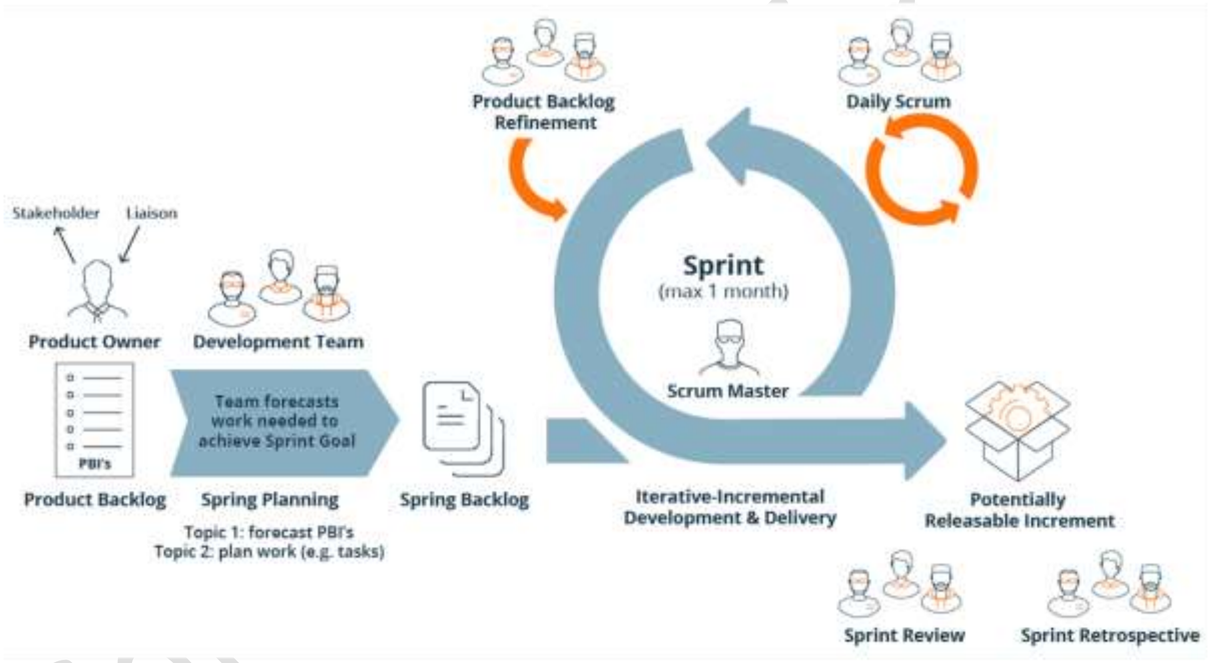
Scrum

Introduction

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

- Process framework: a particular set of practices that must be followed. For example, Scrum process is distinguished by specific concepts and practices - Roles, Artifacts, and Time Boxes.
- Lightweight: overhead of the process is kept as small as possible, to maximize the amount of productive time available for getting useful work done.

Scrum process



- Product Owner creates a product backlog, a list of tasks and requirements the final product needs. The important part is that product backlog must be prioritized.
- The scrum team gets together for sprint planning, which is when the team decides together what to work on first from the product backlog. This subset of items from the product backlog becomes the sprint backlog.
- During the sprint, the team meets to communicate progress and issues, this meeting is called the daily scrum. It is overseen by the scrum master who ensures that all the team members follow scrum's theories, rules, and practices.

- At the end of the sprint, the sprint review meeting is organized by the product owner. During the meeting, the team demonstrates what they completed since the last sprint. At the end of each sprint, the team should have a functioning piece of the product to show for their work.
- The scrum team gathers-up in sprint retrospective meeting, where the team discusses what went well, what did not and if they could have done better.

The cycle repeats for the remaining requirements in the product backlog.

Scrum Roles

There are three distinct roles defined in Scrum - ScrumMaster, the Product Owner, and the Team.

Scrum master, the keeper of the process

- Responsible for making the process run smoothly. Make sure that the goals and scope of the project are understood by everyone involved
- Remove impediments & obstacles that could impact team's productivity.
- Organizing and facilitating the critical meetings.
- Keep information about the Team's progress up to date and visible to all parties.
- Guide the team and product owner to improve the effectiveness of their practices.
- Makes changes to increase the productivity of the scrum team.

Product owner, the keeper of the requirements

- 'Single source of truth' for the team regarding requirements
- Prioritizing the requirements.
- interface between the business (the customers) and the team.
- Managing the product backlog & making sure that it is visible, transparent, and clear.
- Progressing the value of the work done and making rational decisions.
- Collaborating with the development team and stakeholders.

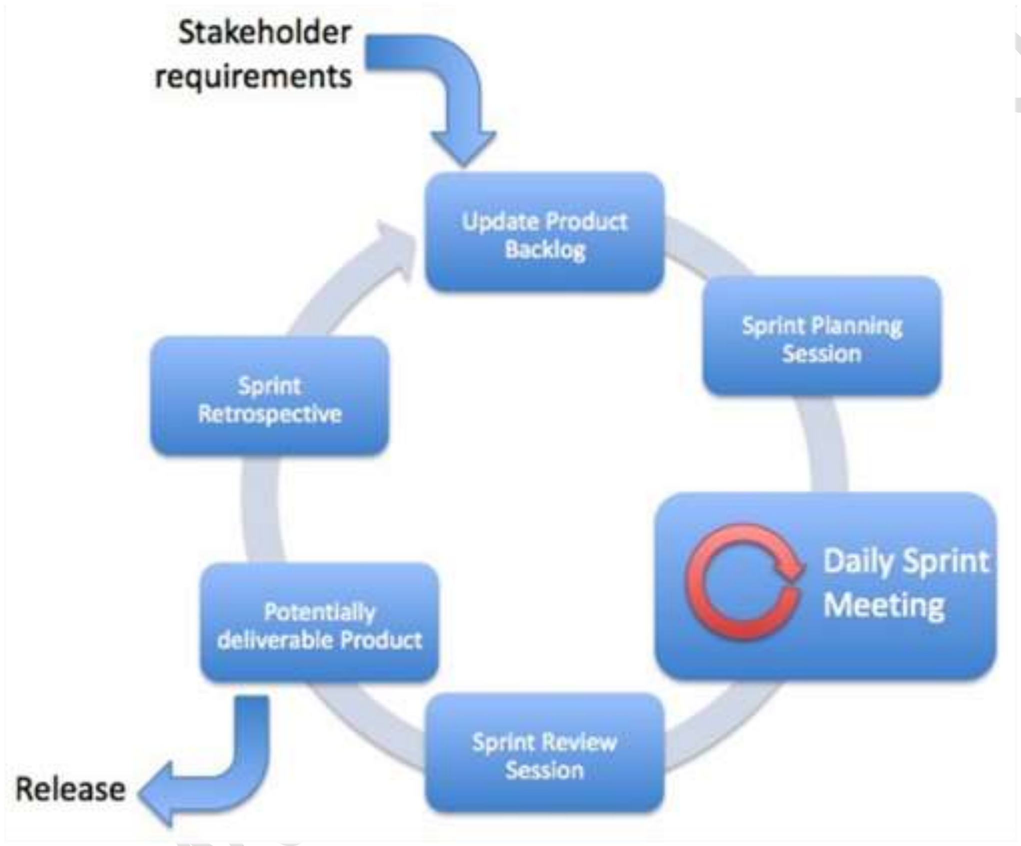
Agile team

- Self-organizing: Team members decide how to break work into tasks, and how to allocate tasks to individuals, throughout the Sprint.
- Cross-functional: a group of people with different functional expertise working toward a common goal.
- Plan the sprint with guidelines from the product owner and scrum master
- Perform sprint execution and adapt the sprint according to changing user requirements. Do the hands-on work of developing and testing the product.
- Help product owner groom the product backlog.
- Inspect and adapt the product & process.

Sprint

Introduction

Sprint is a short, time-boxed iteration to allow the scrum team to focus on delivering value to the customer. During a sprint, the team works to create a done, usable, and potentially releasable product increment. Example, a 2-weeks sprint where a scrum team completes 4 user stories, i.e., ready for review at sprint end.



Timeboxing simply means to limit the maximum time an activity can take. It's the core concept of a Scrum model. Typically, a Sprint duration is 1, 2, 3, or 4 weeks, and this varies from organization to organization. Within this timebox, the Scrum team has to finish the agreed set of work.

Release vs Sprint

The heart of Scrum is a Sprint, a time-box of one week or one month during which a potentially releasable **product increment** (feature or set of features within a product) is created. Release, on the other hand, is the complete feature set for that particular **version** of the product.

Both are considered "shippable" and able to be consumed by the end user, but generally speaking the output of a Sprint is just the short-term part of the end product and cannot stand alone while the output of a Release is the long term, strategic product itself.

Zero Sprint

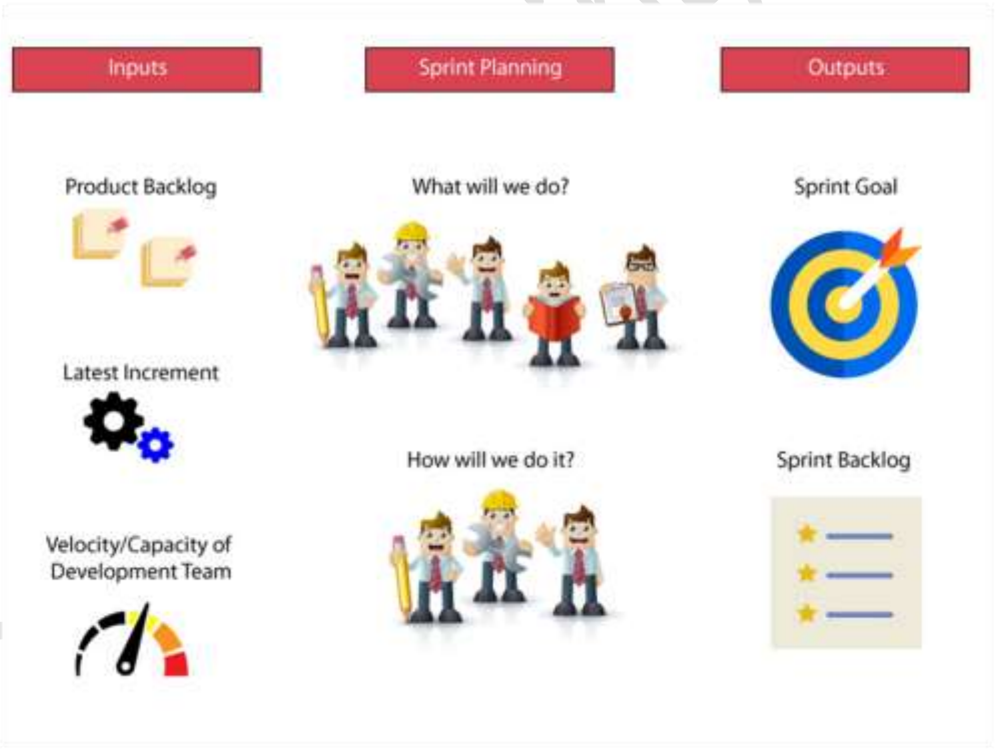
A Sprint 0 is the name often given to a short effort to create a vision and a rough product backlog which allows creating an estimation of a product release. Why? Since there are things that need to be done before a Scrum project can start.

Note: From official scrum guide - there is no Sprint 0. In practical world, when a team sets out to adopt Scrum - usually Sprint 0 is used for the first time to adopt the scrum framework in the current business process.

Scrum Events

Sprint planning

What needs to be delivered in the current Sprint iteration? Every Sprint begins with Sprint planning when the entire team gets together to agree on what needs to deliver in the Sprint (Sprint Goals). The entire team clearly defines deliverables for the Sprint and assigns the work necessary to achieve that goal. Entire scrum team takes part in Sprint Planning to understand the work or tasks to be performed in that Sprint and create a sprint goal.



Note: Sprint planning is timeboxed, maximum of eight hours for a one-month Sprint.
Sprint Goal: Each Sprint needs to have a potentially releasable deliverable. How do we achieve that? By having a clearly defined Sprint goal. The basis of the Sprint goal is the essential features of the product that needs to be built first.

Daily Scrum/Stand Up

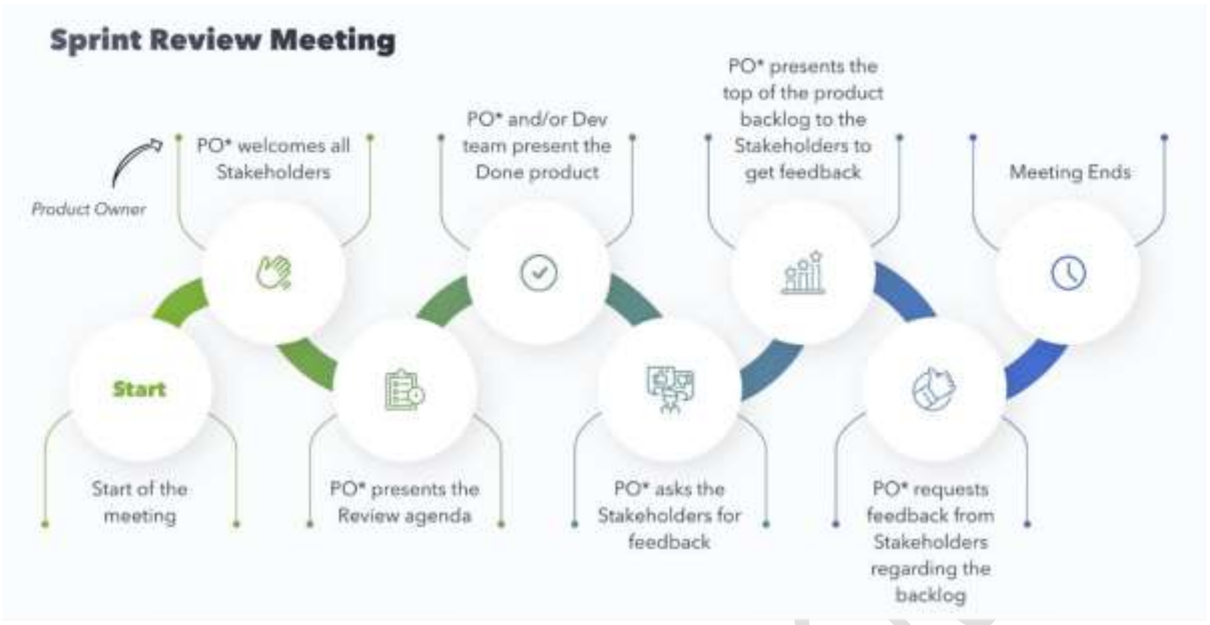
Work done "yesterday"
Work to be done "today"
Any "Impediments"

Also known as daily stand-up, it is a 15-minute daily meeting where the team has a chance to get on the same page and put together a strategy for the next 24 hours. The agenda being,

- What was done yesterday
- What will be done today
- Are there any impediments that are preventing the team from meeting the sprint goal?

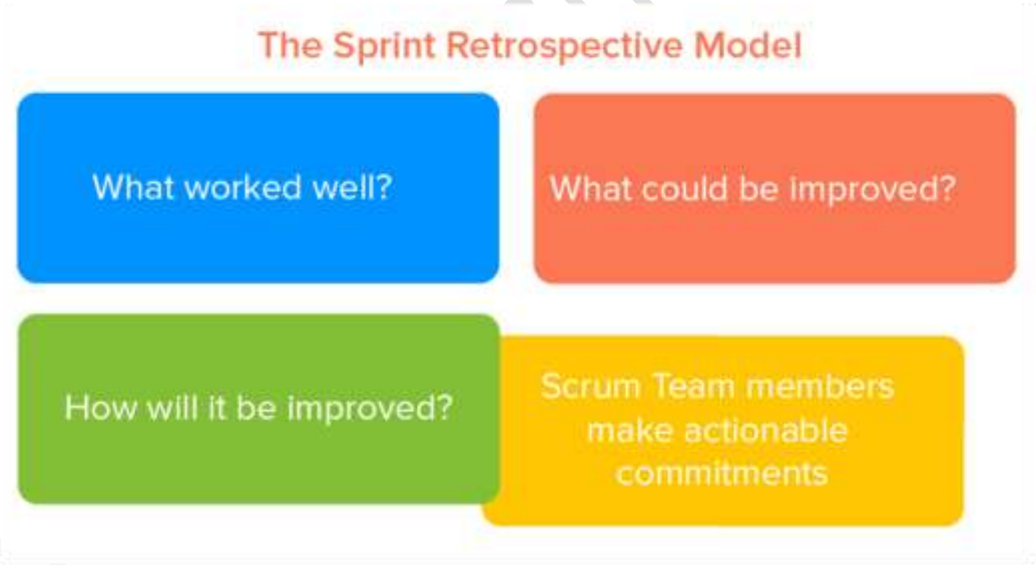
Sprint review

Held at the end of each sprint, the team presents completed work to product owner and stakeholders for approval or any feedback that they might have. Usually, the Product Owner starts by talking about the goals of the Sprint and what functionality is delivered. The Scrum Master can show burn-down charts and the velocity of the team. After which the development team will showcase their work.


Sprint Review Meeting
Product Owner
Start
Start of the meeting
PO* welcomes all Stakeholders
PO* presents the Review agenda
PO* and/or Dev team present the Done product
PO* asks the Stakeholders for feedback
PO* presents the top of the product backlog to the Stakeholders to get feedback
PO* requests feedback from Stakeholders regarding the backlog
Meeting Ends

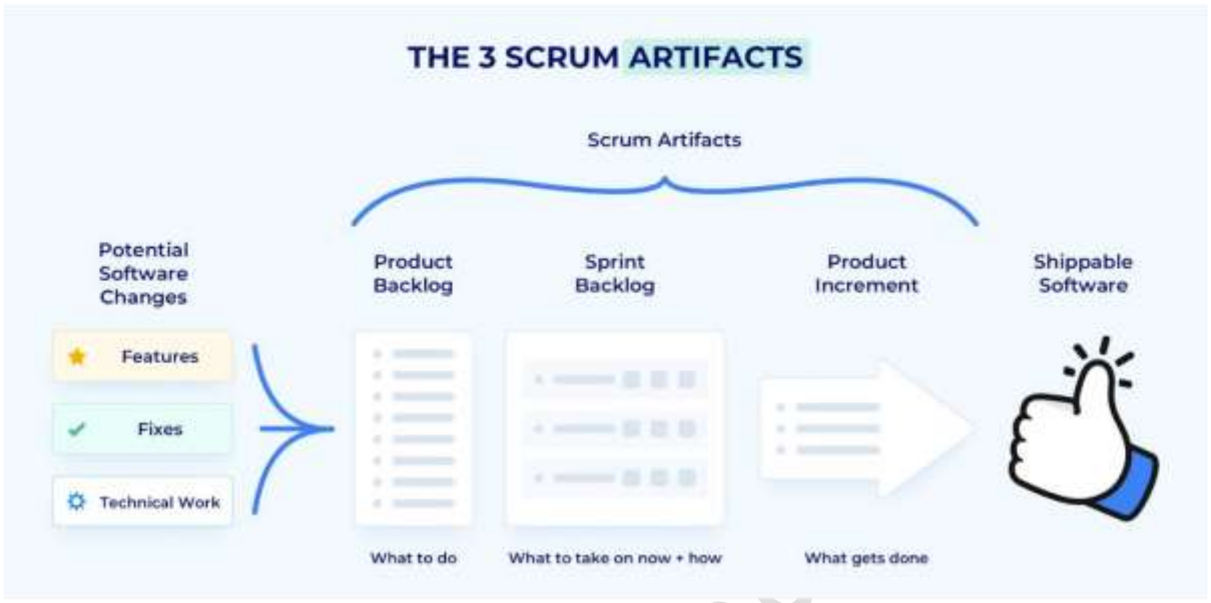
Note: Sprint review is timeboxed. The duration is a maximum of 1 hour for every week of Sprint duration.

Sprint Retrospective


The Sprint Retrospective Model
What worked well?
What could be improved?
How will it be improved?
Scrum Team members make actionable commitments

The team discusses what went right, what went wrong, and how to improve. They decide on how to fix the problems and create a plan for improvements to be enacted during the next sprint. Regardless of how good a Scrum team is, there is always room for improvement.

Scrum artifacts



Product backlog

It's a document that acts as a single source of requirements for everything needed in the product. It includes new features, changes to existing features, bug fixes, infrastructure setups, etc. It is constantly evolving and is never complete. The Product Owner manages the Product Backlog, including how it's made available to the scrum team, its content, and how it's ordered and prioritized, everything. The Product Owner and the rest of the scrum team work together to review the Product Backlog and make adjustments as and when necessary.

Each requirement in the product backlog have a,

- Description
- Order based on priority
- Estimate
- Value to the business

Sprint backlog, Sprint goal

The list of all items from the product backlog that need to be worked on during a sprint. A real-time picture of the work that the team currently plans to complete during the sprint.

- It is the outcome of sprint planning meeting sessions
- The development team owns the sprint backlog and divides tasks according to their skills
- It is a highly visible, real-time picture of the work that the Development Team plans to accomplish
- The sprint backlog is dynamic in nature because each sprint has repeated changes to reach the goal.

Product Increment

"*What gets produced at the end of a development period or timebox*".

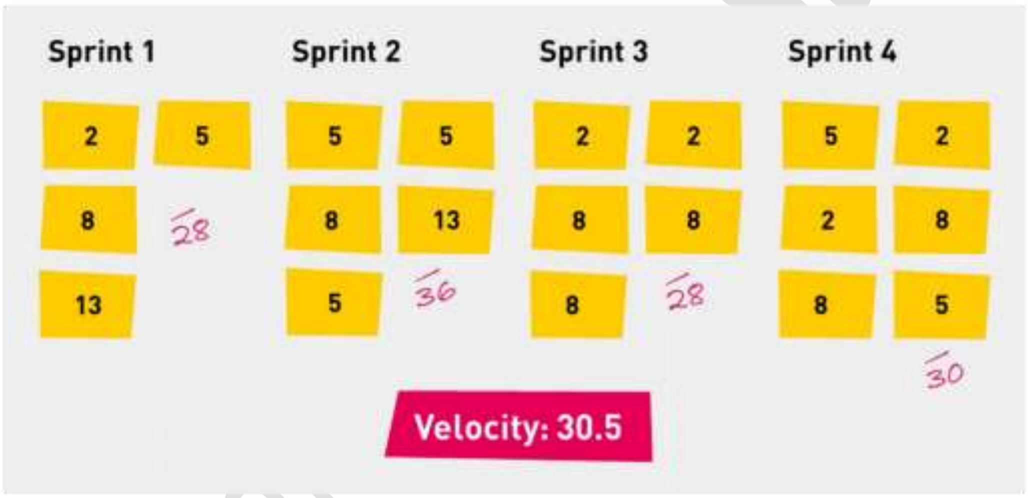
The most important artifact is the product increment, or in other words, the sum of product work completed during a Sprint, combined with all work completed during previous sprints. In a Scrum environment, the team would have a 'definition of done', so that they know that the product increment is complete.

Agile metrics

Velocity

Agile Velocity is an extremely simple method for measuring the rate at which scrum teams consistently deliver business value. In other words - How much product backlog effort a team can handle in one sprint? It's the rate at which a team delivers stories from the product backlog, i.e., sum of estimates of delivered (i.e., accepted) features per iteration. It can be measured in story points, days, ideal days, or hours that the Scrum team delivers – all of which are considered acceptable.

A team delivers 20 and 30 story points in the first & second iteration respectively. The average velocity = Sum of Story points delivered / Number of iterations = (20 + 30)/2 = 25 Story points!



- Only the aggregate velocity of the team matters, and the phrase "individual velocity" is meaningless.
- Velocity is NOT productivity. Instead of a speedometer, it is more like a measurement of a team's rhythm and should be used to monitor team health.

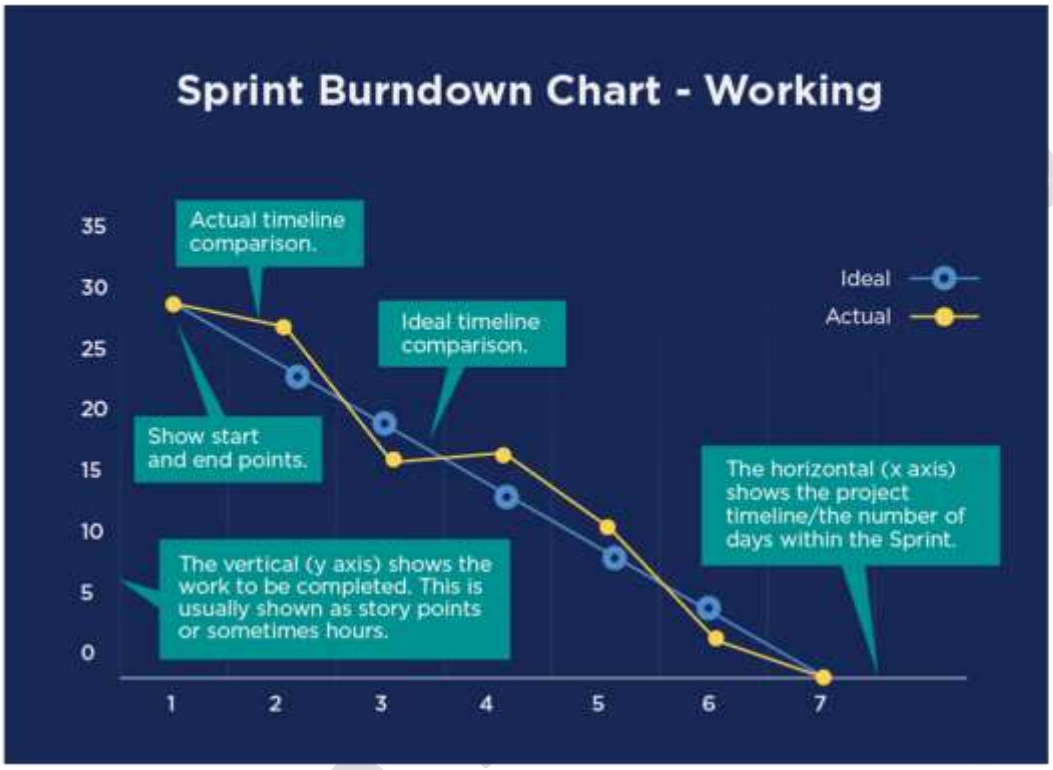
The goal is not maximized velocity, but rather optimal velocity over time, which takes into account many factors including the quality of the end product. Benefits,

- Reporting, i.e., how much value is delivered (Story points/User stories)
- Resource and project planning (timelines)
- Adjusting project scope and business priorities, and forecasting release dates
- Identifying and removing obstacles that affect velocity

Do not try to over-complicate velocity – it really is a straight forward concept and a great deal of its value lies in its inherent simplicity.

Burndown chart

Burndown and burnup charts are two types of charts that project managers use to track and communicate the progress of their projects. A burn down chart shows **how much work is remaining** to be done in the project. Its purpose is to enable that the project is on the track to deliver the expected solution within the desired schedule.



A graphical representation of work left to do versus time. You can define remaining work based on Stories or Tasks. The ideal line is going down in a straight line from top left to down right. This indicates a healthy project and a well-functioning Scrum team. Value is being delivered constantly in a linear fashion.

Burnup chart

A burn up chart shows **how much work has been completed**, and the total amount of work. There are two main lines shown on the chart: one for the total project work planned, and the other for tracking the work completed to date.

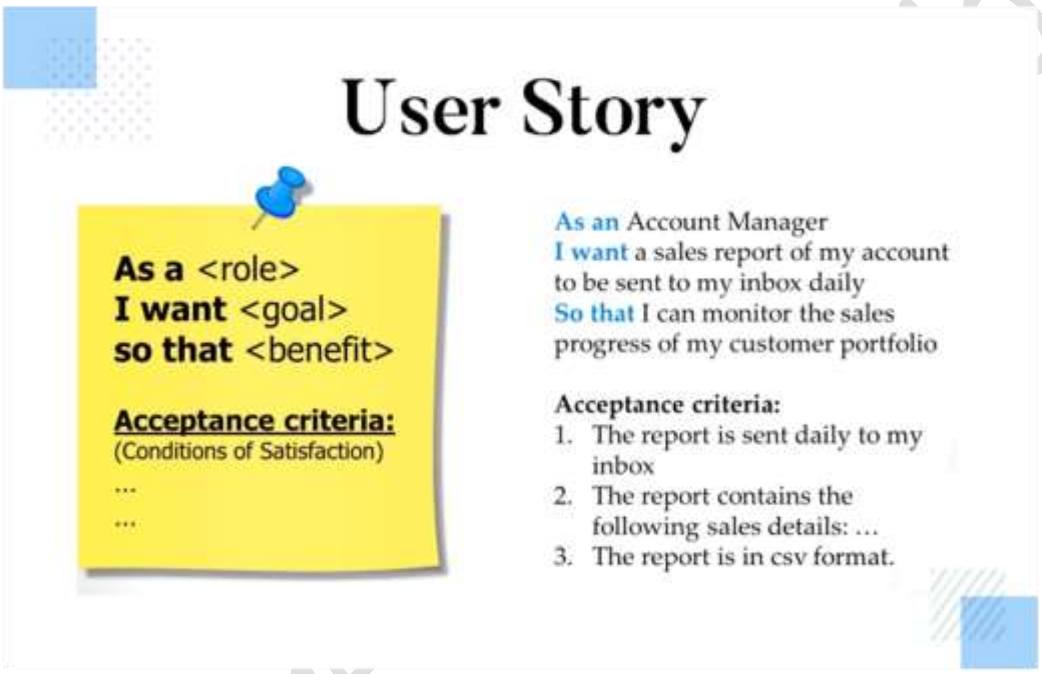
By comparing the work, a team has accomplished so far with the total amount of work planned, it allows project managers and teams to quickly see how their workload is progressing and whether project completion is on schedule.

Agile Requirements

Epic, User Story & Tasks

Epic: a big story that will help the end-user to resolve a business problem.

User Story: smaller pieces of functionality, and they directly map to an epic. A narrative text(s) that describe an interaction of the user and the system, focusing on the value a user gains from the system.



User Story

As a <role>
I want <goal>
so that <benefit>

Acceptance criteria:
(Conditions of Satisfaction)
...
...

As an Account Manager
I want a sales report of my account to be sent to my inbox daily
So that I can monitor the sales progress of my customer portfolio

Acceptance criteria:
1. The report is sent daily to my inbox
2. The report contains the following sales details: ...
3. The report is in csv format.

As a < user > I want to < goal > so that < acceptance criteria >

Note: Scrum doesn't say that the requirements need to be in the story format. As per Scrum, you can have the requirements in any form, though the story is the most popular format to document requirements.

Tasks: actual work that needs to be done to complete a story. A Story is divided into tasks, and once all these tasks finish, we can mark the story as "Done" or "Completed." E.g., create a UI form, develop API, create a backend database table, etc.

Good user story, INVEST

The acronym INVEST helps to remember a widely accepted set of criteria, or checklist, to assess the quality of a user story.

A good user story should be:

- "I" ndependent (of all others)
- "N" egotiable (not a specific contract for features)
- "V" aluable (or vertical)
- "E" stimable (to a good approximation)
- "S" mall (so as to fit within an iteration)
- "T" estable (in principle, even if there isn't a test for it yet)

Each user story is expected to yield, once implemented, a contribution to the value of the overall product, irrespective of the order of implementation; these and other assumptions are captured by the INVEST formula.

Acceptance criteria

Acceptance Criteria are a set of statements, each with a clear pass/fail result, that specify both functional and non-functional requirements, and are applicable at the Epic, Feature, and Story Level. Acceptance criteria constitute our "Definition of Done", i.e., when a work item is complete and working as expected. The Given/When/Then format is helpful way to specify criteria:

Given some precondition When I do some action Then I expect some result.

In simple terms, acceptance criteria refer to a set of predefined requirements that must be met to mark a user story complete. It clearly defines the scope, desired outcomes of, and testing criteria for pieces of functionality that the delivery team is working on.

Note: Often Acceptance Criteria are defined by the Product Owner (PO) when creating a Product Backlog Item and then adapted and extended in the Backlog Refinement.

Definition of Done (DoD)

Usually, a checklist of all the work that team needs to do before the team can call the user story/product increment as “Done”.

The Definition of Done (DoD) must be clear to all the stakeholders. It ensures that when the Scrum team calls out a product increment as “Done”, everyone has the same shared understanding of what this means. Example,

- Code is written.
- Code is documented.
- Code review has been completed.
- Unit test passed
- Build has been made and deployed on a testing environment.
- Acceptance criteria for each issue met
- Functional tests passed
- Non-functional requirements met
- Test automation scripts are developed & executed
- Product owner accepts the User Story

Note: Scrum team defines the definition of done. They are the ones who will be accountable to meet this definition, so it’s essential that the team creates it and agrees to it.

Agile Estimation & planning

Estimation

Agile estimation is about evaluating the effort required to complete each work item listed in the prioritized backlog, which, in turn, helps in better sprint planning. Most Agile estimation techniques use relative units. This means that we don't try to estimate dollars or days directly. Instead, we use "points" or even qualitative labels and simply compare the items we are estimating to each other.

0
0.5
1
2
3
5
8
13
20,40
100, ∞

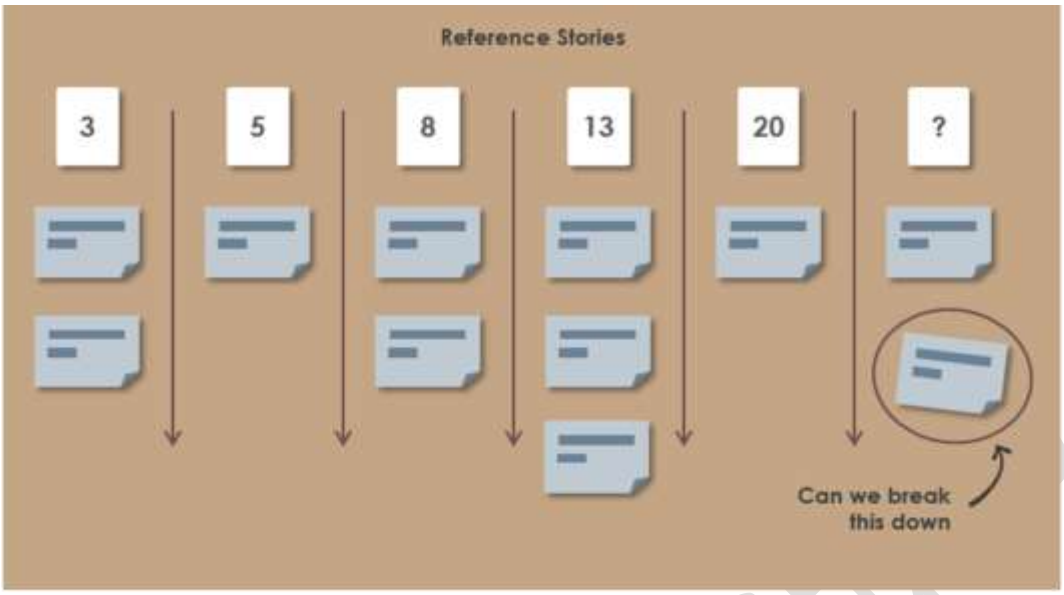
Agile estimations are essential for:

- Making teams accountable for deliverables
- Inducing discipline across the Agile team
- Predicting the approximate time, it will take to finish a project
- Enabling better sprint management
- Improving team productivity

Note: Estimation is done by the entire team during Sprint Planning Meeting. The objective of the estimation would be to consider the User Stories for the sprint by priority and by the ability of the team to deliver during the time box of the sprint.

Story points

Traditional software teams give estimates in a time format: days, weeks, months. Many agile teams, however, have transitioned to story points. Story points are **units of measure for expressing an estimate of the overall effort** required to fully implement a product backlog item or any other piece of work. Teams assign story points relative to work complexity, the amount of work, and risk or uncertainty.



Note: Story points go wrong when they're used to judge people, assign detailed timelines and resources, and when they're mistaken for a measure of productivity. Instead, teams should use story points to understand the size of the work and the prioritization of the work.

Estimation Techniques

Planning poker

Planning poker makes use of story points to estimate the difficulty of the task at hand. Based on the Fibonacci sequence, the story point values that can be assigned are 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100. Each of these represent a different level of complexity.

When the discussions around user story are finished, team members will select a card with the story point they believe needs to be assigned to the user story. Further discussions will ensue until there is a consensus.

The Bucket System

Using the same sequence as Planning Poker, a group or a team estimate items by placing them in "buckets". The Bucket System is a much faster because there is a "divide-and-conquer" phase.

Big/Uncertain/Small

For super-fast Agile estimation, the items to be estimated are simply placed by the group in one of three categories: big, uncertain and small. The group starts by discussing a few together, and then, like the Bucket System, uses divide-and-conquer to go through the rest of the items.

T-Shirt Sizes

Items are categorized into t-shirt sizes: XS, S, M, L, XL. The sizes can, if needed, be given numerical values after the estimation is done. This is a very informal technique, and can be used quickly with a large number of items.

Ordering Protocol

Items are placed in a random order on a scale labelled simply "low" to "high". Each person participating takes turns making a "move". A move involves one of the following actions: change the position of an item by one spot lower or one spot higher, talking about an item, or passing. If everyone passes, the ordering is done.

Other agile methods & frameworks

Extreme programming

An agile software development framework that is the most specific regarding appropriate engineering practices for software development.

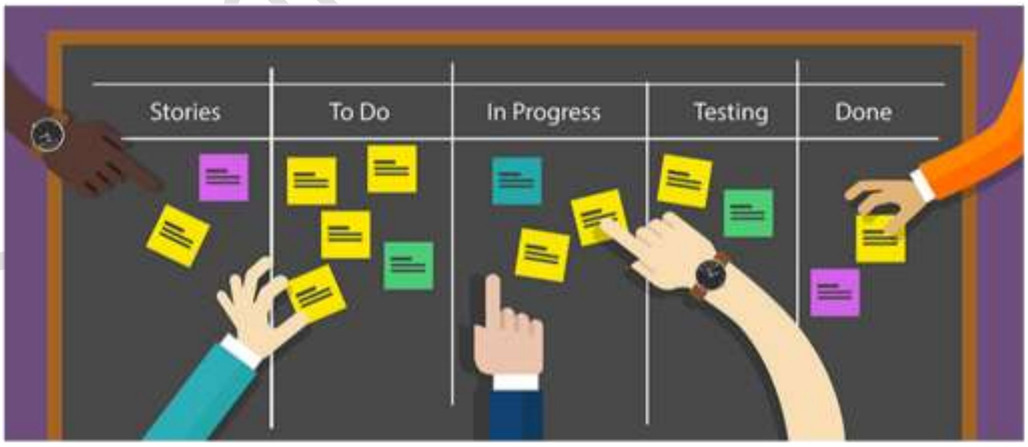
- Dynamically changing software requirements
- Risks caused by fixed time projects using new technology
- Small, co-located extended development team
- The technology you are using allows for automated unit and functional tests

The core of XP is the interconnected set of software development practices listed below,

- The Planning Game, Small Releases, Metaphor, Simple Design, Testing, Refactoring
- Pair Programming: all production software is developed by two people sitting at the same machine.
- Collective Ownership
- Continuous Integration: a practice where code changes are immediately tested when they are added to a larger code base.
- 40-hour week, On-site Customer, Coding Standard

Kanban

Kanban is a popular framework used to implement agile. Work items are represented visually on a Kanban board, allowing team members to see the state of every piece of work at any time.

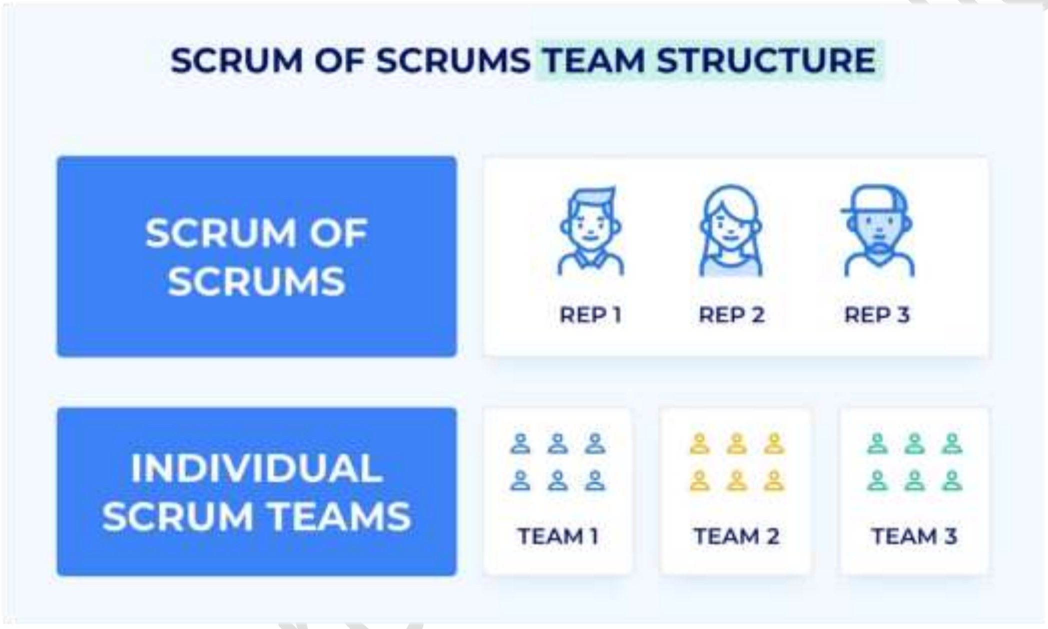


For example, you might create lists for “Backlog,” “Up Next,” “In Progress,” and “Done!” Each task is organized as a card, which you move across the lists as they are queued up, worked on, and completed.

Scrum - of – Scrums

“Scrum of Scrums” is a scaled agile technique that offers a way to connect multiple scrum teams. Think of it as multiple smaller scrum teams who are coordinating to deliver a complex product. The key here is to enable effective & efficient coordination between smaller scrum teams to deliver full value.

A cross-team synchronization method used in case when multiple teams are involved, to support the agile teams in collaborating and coordinating their work with other teams. This involves conducting a scrum meeting of reps from individual scrum teams, additional roles like Chief product owner, SOS master, etc.

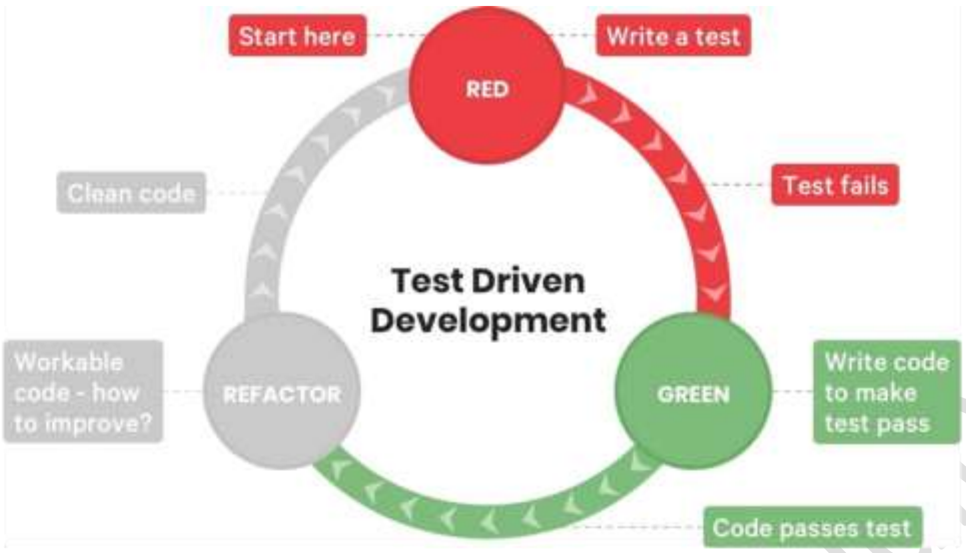


Example, say we have 3 separate scrum teams of 5 members each. Scrum of Scrums meeting will enable reps from these 3 teams to coordinate work. It's "Scrum of Scrums". Spot the difference 'Scrum' of 'Scrums'.

Test-driven Development

A software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only.

Each iteration starts with a set of tests written for a new piece of functionality. These tests are supposed to fail during the start of iteration as there will be no application code corresponding to the tests. In the next phase of the iteration Application code is written with an intention to pass all the tests written earlier in the iteration. Once the application code is ready tests are run.



Any failures in the test run are marked and more Application code is written/re-factored to make these tests pass. Once application code is added/re-factored the tests are run again. This cycle keeps on happening till all the tests pass. Once all the tests pass, we can be sure that all the features for which tests were written have been developed.

SAFe

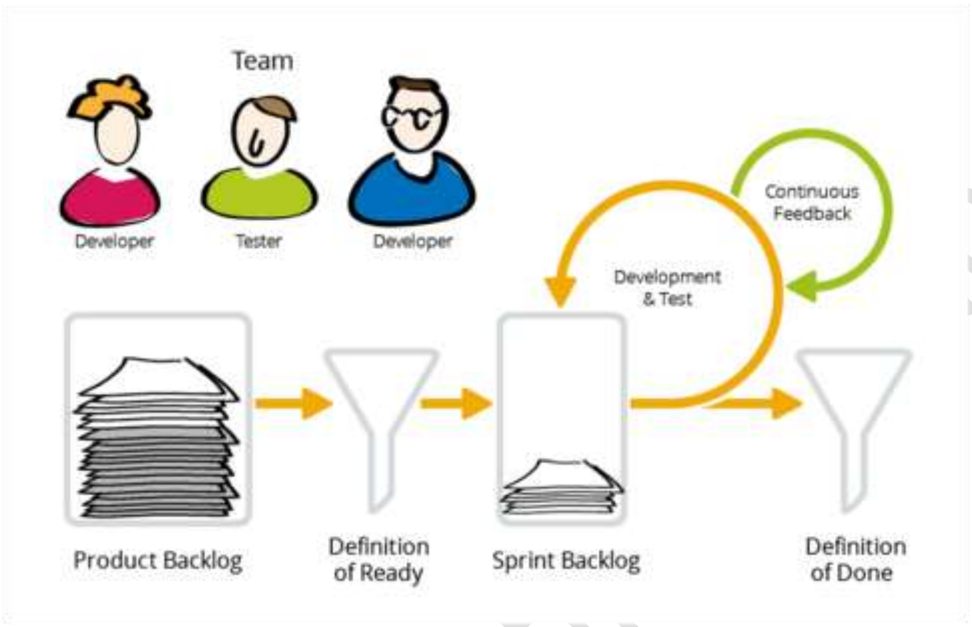
Scaled agile framework (SAFe) - a set of organizational patterns to guide enterprises in scaling lean and agile practices beyond a single team. i.e., if you want to implement standard agile practices across projects & teams within an organization.

Scrum-ban

Project management framework that combines important features of two popular agile methodologies: Scrum and Kanban. Scrumban merges the structure and predictability of Scrum with Kanban's flexibility and continuous workflow. It embraces on-demand planning. It involves applying Kanban principles - visualization of workflow, and flexible processes—to a team's Scrum framework. Scrumban teams also use Kanban processes, such as the pull system, which provides a continuous workflow. That is, tasks are pulled into the doing column when the team is ready to execute.

Agile Testing

A testing practice that follows the rules and principles of agile. Unlike Waterfall method, Agile testing can begin at the start of the project with continuous integration.

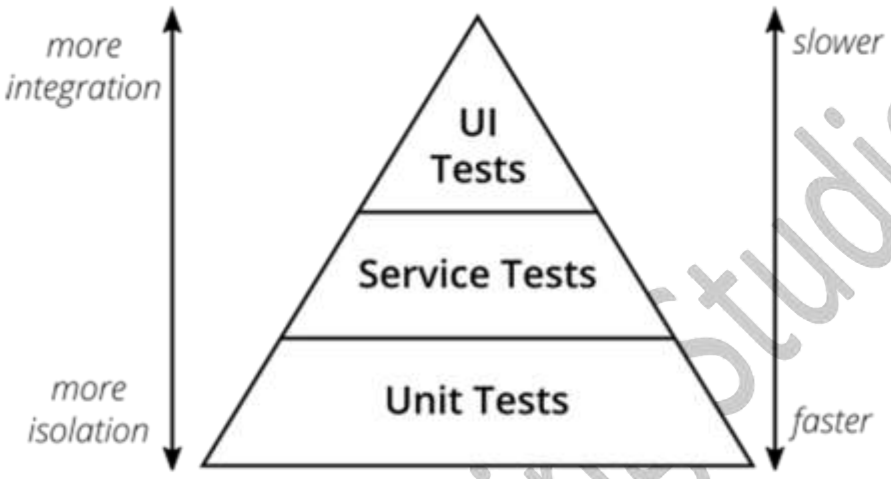


Agile Testing is not sequential like Waterfall but continuous, i.e., testing happens in parallel with feature development as part of every sprint - goes hand in hand with development work and provides an ongoing feedback loop into the quality.

It is a collaborative effort between testers, developers, product owners and even customers.

Testing pyramid

The "Test Pyramid" is a metaphor that tells us to group software tests into buckets of different granularity. It also gives an idea of how many tests we should have in each of these groups.



The assumption here is that automated unit tests are cheap, easy, fast to run and isolated, compared to those slow, brittle, hard to write end-to-end tests that require a full working system and a web browser or mobile device. The foundation of a test effort should be unit tests, with fewer service tests and very few end-to-end tests, creating a bit of a pyramid.