Software Engineering

Software Engineering is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.

Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product. Engineering on the other hand, is all about developing products, using well defined, scientific principles and method

IEEE defines software engineering as:

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

Fritz Bauer, a German computer scientist, defines software engineering as: “Software engineering is the establishment and use

of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.”

Need of Software Engineering The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working. Following are some of the needs stated:

- Large software - It is easier to build a wall than a house or building, likewise, as the size of the software becomes large, engineering has to step to give it a scientific process.
- Scalability- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
- Cost- As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But, cost of the software remains high if proper process is not adapted.
- Dynamic Nature- Always growing and adapting nature of the software hugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where the software engineering plays a good role.
- Quality Management- Better process of software development provides better and quality software product.

A formal definition of software engineering might sound something like, “An organized, analytical approach to the design, development, use, and maintenance of software.” More intuitively, software engineering is everything you need to do to produce successful software. It includes the steps that take a raw, possibly nebulous idea and turn it into a powerful and intuitive application that can be enhanced to meet changing customer needs for years to come.

Software development Life cycle

SDLC

Every software development process has a life cycle known as Software development life cycle.This life cycle has different stages

Planning Stage :-

Planning stage involves creating scope of application ,financial as well as technical, how many people to involve ,skills required, budget for entire project ,duration in which all the process has to be completed and software to be handed over to the client

At the planning stage itself certain key issues are discussed with the team

- Objective of intended software system
- different stake holders have to be identified ,
- role of each person involved with the process of development ,
- financial limitations with the team and customer
- Business risk involved

Planning process can include discussion with stake holders ,and team that is going to develop the software

[

Who Is Considered a Stakeholder?

While every development project is unique, there are some universal categories that can be used to guide stakeholder identification.

End users and beneficiaries

These are the people who will be most directly affected by the software. Their buy-in is essential. No matter how flashy or efficient software is, if end users don't like it they won't use it. End users fall into **three main groups:**

Direct users

Those who will use the software directly are usually most concerned with how it will fit into their current workflows. They

want to know that it solves a significant problem or otherwise makes their job easier.

Secondary users

Direct users interact with the software itself. Secondary users rely on the products of the software. New software needs to produce results in a format that fits into secondary users' workflows. Forgetting about this group can cause one problem while solving another, like suddenly generating reports in a format secondary users can't **integrate into their analytics**.

Beneficiaries

These are all the people affected by the software's products. The term encompasses a huge base of customers and vendors who focus more on results than process. Their input should revolve around the services or information the software will provide.

]

Analysis stage:-

At this stage all the requirements related to develop the software are gathered ,analyzed and discussed with team and stake holders

Gathering requirements ,involves

talking to stake holders and understanding complete requirement of users

Consultation can be made with a domain expert of different people involved currently in the existing process

A complete analysis and feasibility report ,metrics and analysis documents form critical part of development process ,as they can be accessed anytime to take a design decision, check deviation from intended objective , analyze what is possible within available resources and what exactly the customer wants

Design Phase :=
This phase involves design the actual system that needs to be developed .Designing can involve different methodologies like creating a map of the system ,use cases ,UML Diagrams ,developing prototypes .
The purpose of this phase is to ensure that all requirements analyzed during requirement analysis phase are finally filtered and put on paper ,or prototyped and the team can visualize the final outcome that they will get once the development process is over

Development phase:

In this phase ,the software that has been designed is put to code .Different programming languages and technologies can be used for coding ,This depends on methodology to be used

for programming .Two popular methodologies are procedural and Object Oriented programming
Most of the time it is decided at the design stage itself ,which methodology has to be used to code for intended software system.
Systems are mostly developed today around Object Oriented Programming

Different programming languages that target Object Oriented Programming are
C++,Java,C#

Testing and Integration phase :
This phase involves testing of system developed.The objective of testing is to identify bugs and issues in the system.Is the intended system as per requirements or has it deviated from the intended purpose .
The intention of testing is not to stall the development process ,or negate the system built ,but to identify bugs that can damage the hole process of development .
The tesing process has it's own lifecycle called software testing life cycle

Deployment and Maintenance :

The system developed through the software development process has to be delivered to the client ,either as a package if it is a desktop application ,or through different stores ,if it is a mobile application or through web it is a web application

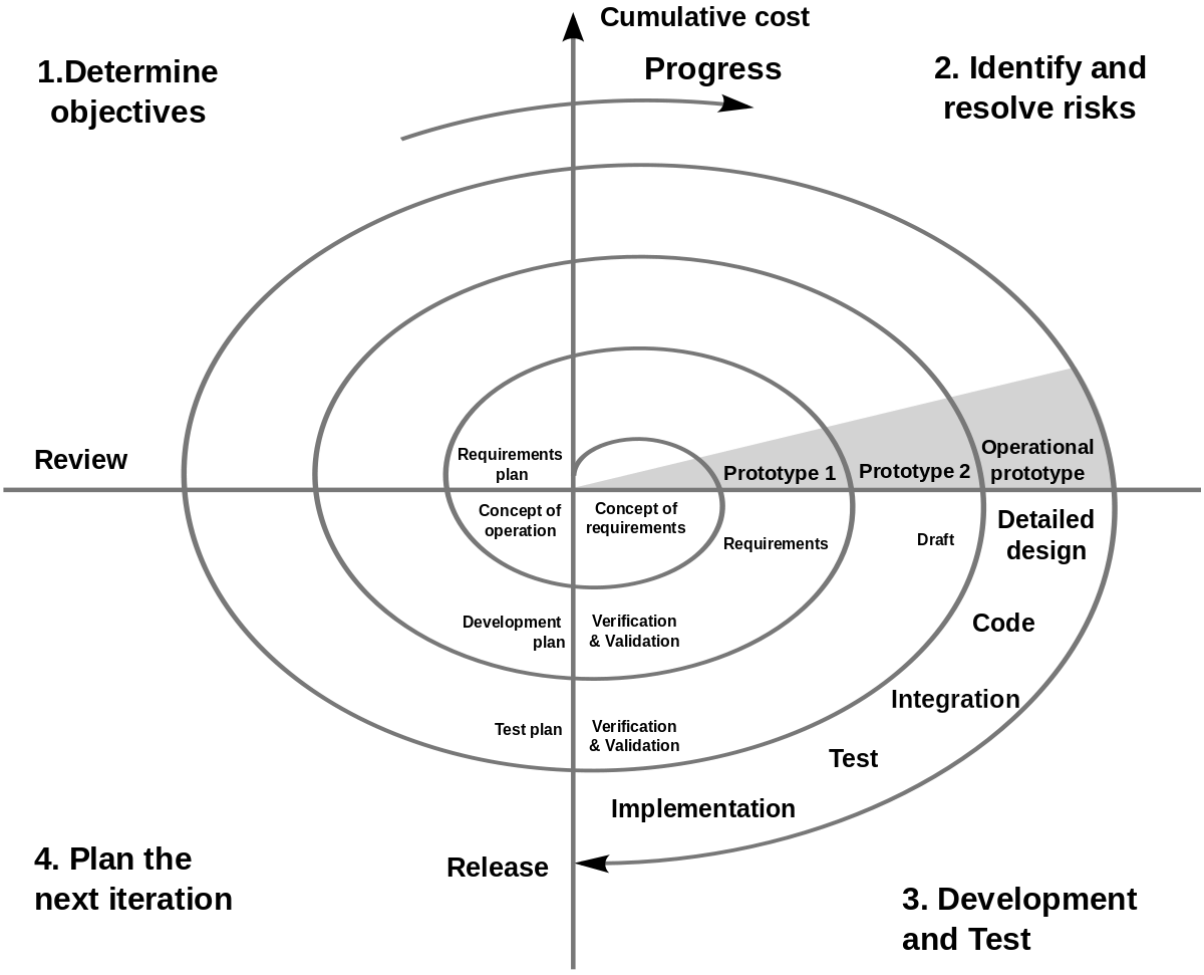
Maintenance involves different things like

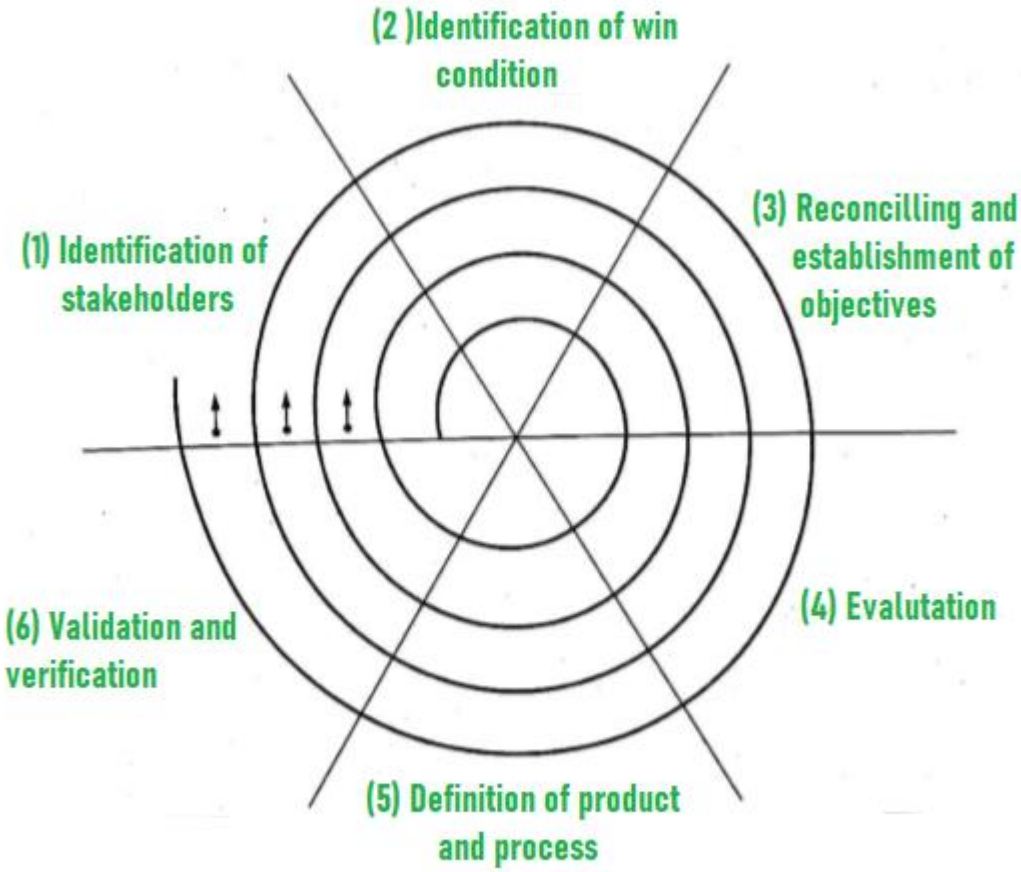
- updating software from time to time
- providing patches to plug undetected bugs
- cleaning up unwanted data collected through interaction of user with software

- monitoring software for security related risks
- scaling application to meet increasing customer demand
- enhancing security needs

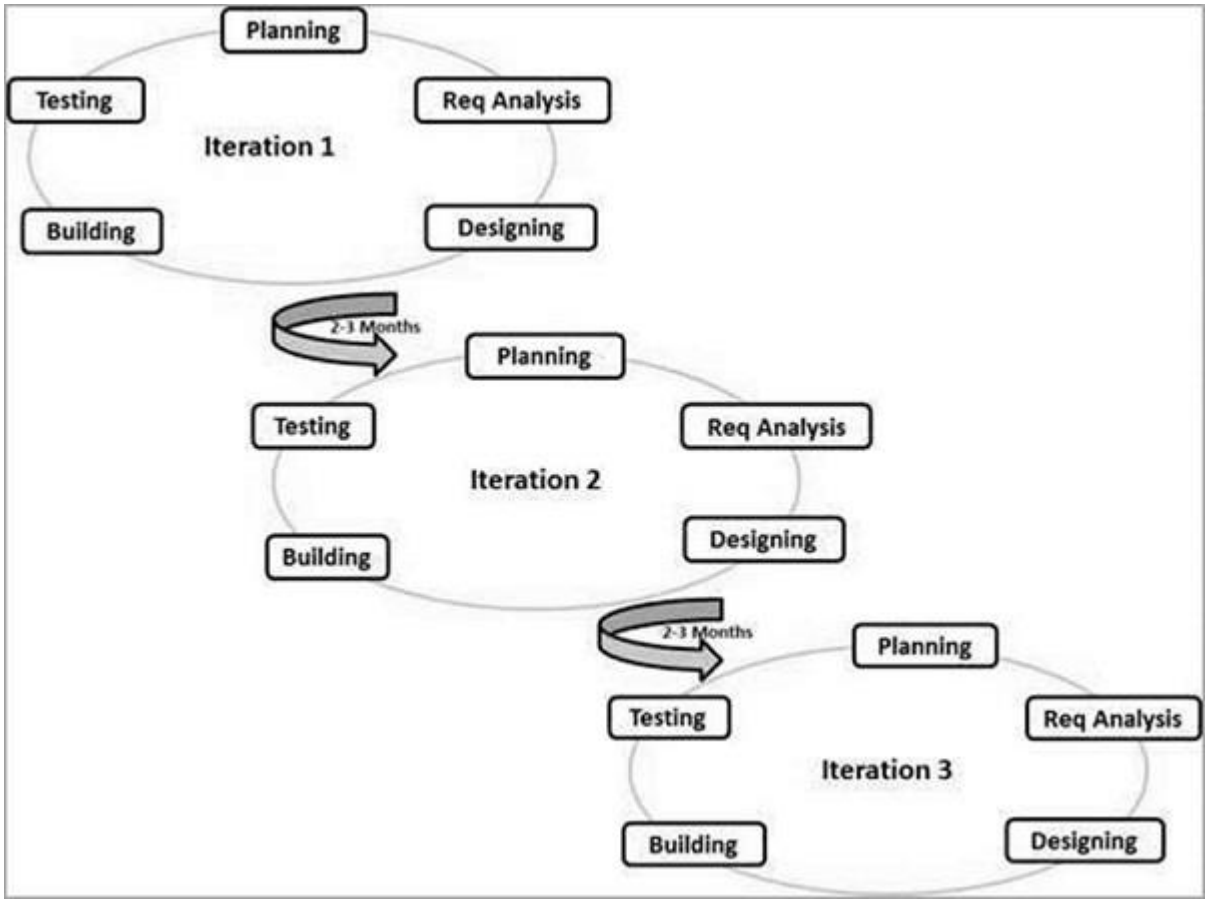
The six stages of software development appear based on type of model selected for development

different models that can be used to implement are Waterfall model, Spiral Model ,Win-Win Spiral modal ,agile development


Cumulative cost
Progress
1.Determine objectives
2. Identify and resolve risks
Review
Requirements plan
Concept of operation
Concept of requirements
Prototype 1
Prototype 2
Operational prototype
Requirements
Draft
Detailed design
Development plan
Verification & Validation
Code
Integration
Test plan
Verification & Validation
Test
Implementation
Release
4. Plan the next iteration
3. Development and Test



WIN-WIN Spiral Model


Planning
Testing
Req Analysis
Iteration 1
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 2
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 3
Building
Designing

AGILE

Agile is a software development methodology to build a software incrementally using short iterations of 1 to 4 weeks so that the development process is aligned with the changing business needs. Instead of a single-pass development of 6 to 18 months where all the requirements and risks are predicted upfront, Agile adopts a process of frequent feedback where a workable product is delivered after 1 to 4-week iteration.

Different Agile Methodologies

1) Kanban

Originating from the Japanese language, the translation of the word ‘Kanban’ is “visual board or signboard” and is connected to the concept of “just in time”! Initially, the Kanban concept was introduced as a lean manufacturing system and slowly drove its way to agile software development teams. This method uses visual methods for developing and managing projects.

Projects through Kanban are overseen with the help of the Kanban Board, which is divided into columns to depict the process flow of the software development. This helps in increasing the visibility of teams as the teams can see the progress through every stage of development and prepare for the upcoming tasks to deliver the product “just in time”!

This method requires thorough interaction and transparency to enable the team members to be equipped with the right stage of development at any time and have a cohesive flow of work at all times.

2) Scrum

One of the most popular agile methodology examples is the agile scrum development methodology, which is depicted by various cycles of development. Similar to Kanban, Scrum breaks down the development phases into stages or cycles called ‘sprints’. The development time for each sprint is maximized and dedicated, thereby managing only one sprint at a time.

Scrum and agile methodologies focus on continuous deliverables, and thus this method lets designers adjust priorities to ensure that any incomplete or overdue sprints get more attention.

Scrum Team has exclusive project roles such as a scrum master and a product owner with constant communications on the daily scrum where the activities are harmonized to devise the best way to implement the sprint.

3) Extreme Programming (XP)

Extreme Programming (XP) is a methodology that emphasizes teamwork, communication, and feedback. It focuses on constant development and customer satisfaction. Similar to scrum, this method also uses sprints or short development cycles. This is developed by a team to create a productive and highly efficient environment.

The extreme Programming technique is very supportive in a situation of constant and varying demands from the customers. It motivates the developers to accept changes in the customer’s demands, even if they pop up in an advanced phase of the development process.

In Extreme Programming, the project is tested from the initial stages by collecting feedback that progresses the output of the system. This also presents a spot check to implement easily any customer requirements

4) Crystal

Introduced by Mr Alistair Cockburn, one of the monumental persons in formulating the Agile manifesto for software development, Crystal is a group of smaller agile development methodologies comprising Crystal Yellow, Crystal Clear, Crystal Red, Crystal Orange, and more. Each has its peculiar and exclusive framework that is characterized by factors such as system criticality, team size, and project priorities. Depending on the nature of the project or system criticality such as Comfort (C), Essential Money (E), Discretionary Money (D), and Life (L), the kind of crystal agile methodology is chosen.

Similar to other methodologies of Agile, Crystal also addresses prompt delivery of software, regularity, less administration with high involvement of users, and customer satisfaction. The Crystal family advocates that each system or project is inimitable and necessitates the solicitation of diverse practices, processes, and policies to achieve the best results, earning the name of the most lightweight methods of agile methodology.

5) Dynamic Systems Development Method (DSDM)

To address the need for a standard industry charter for the swift delivery of software, the Dynamic Systems Development Method (DSDM) was developed. DSDM gives a comprehensive structure that is defined and modified to create a plan, execute, manage, and scale the procedure of software development. Based on a business-driven approach and eight principles, the DSDM believes that modifications to the project are always expected, and quality with timely delivery must never be negotiated.

6) Feature-Driven Development (FDD)

Several industry-recognized best practices are incorporated into this iterative, customer-centric, and incremental agile method. Its primary goal is to consistently produce working software in a timely fashion.

Lifecycle stages include developing an overarching model of the project; creating feature lists; planning by feature; designing by feature; and finally building by feature. Using this five-step process, large project teams will be able to move their products forward at a steady pace.

7) Lean Software Development

This agile methodology is based on seven principles:

- Deleting what doesn't matter- Anything that doesn't add value is removed from the project
- Quality development- The discipline and control of the number of residuals created are essential to quality development
- Knowledge creation- The team is driven to document the entire infrastructure to preserve this value in the future
- Defer commitments- This point encourages the team to focus less on planning and anticipating ideas without first having a prior and complete understanding of the business requirements
- Delivery promptly- Providing value to the customer as quickly as possible
- Respecting the team- two essential points are communication and conflict management
- Optimize the whole- To create a flow of true value, the development sequence must be perfected enough to remove errors from the code
  Using this lean methodology, development time and resources are optimized. This method is easily scalable and adaptable to projects of any size.

8) Scaled Agile Framework (SAFe)

A set of workflow and organizational patterns for implementing agile practices at an enterprise scale is known as the Scaled Agile Framework (SAFe). Adopting SAFe allows you to take advantage of a framework that is relatively light while still maintaining the centralized decision-making required at the enterprise level for software development efficiency. To put it another way, SAFe takes the agile philosophy and applies it to software executives who are tasked with addressing more strategic issues.

SCRUM Methodology

Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

Scrum is a process framework that has been used to manage complex product development since the early 1990s. Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. Scrum makes clear the relative efficacy of your product management and development practices so that you can improve.

The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.

The rules of Scrum bind together the events, roles, and artifacts, governing the relationships and interaction between them.

Roles in Agile

Scrum Master A Scrum Master is a team leader and facilitator who helps the team members to follow agile practices so that they can meet their commitments.

The responsibilities of a scrum master are as follows:

- To enable close co-operation between all roles and functions.
- To remove any blocks.
- To shield the team from any disturbances.

To work with the organization to track the progress and processes of the company.

- To ensure that Agile Inspect & Adapt processes are leveraged properly which includes o Daily stand-ups, o Planned meetings, o Demo, o Review, o Retrospective Meetings, and o To facilitate team meetings and decision-making process.

Product Owner A Product Owner is the one who drives the product from business perspective.

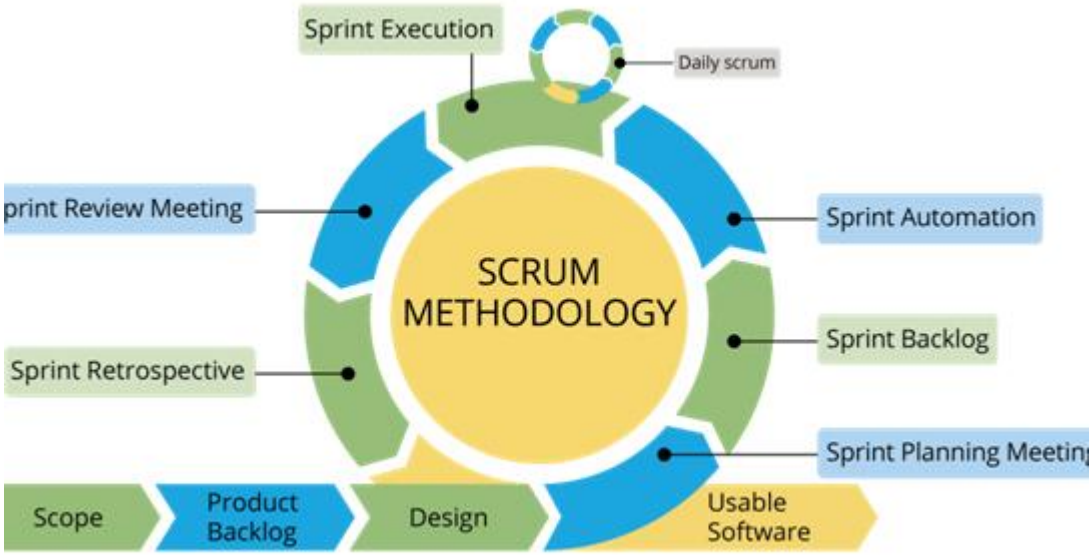
The responsibilities or a Product Owner are as follows:

- To define the requirements and prioritize their values.
- To determine the release date and contents.
- To take an active role in iteration planning and release planning meetings.
- To ensure that team is working on the most valued requirement.
- To represent the voice of the customer.
- To accept the user stories that meet the definition of done and defined acceptance criteria.

Cross-functional Team

Every agile team should be a self-sufficient team with 5 to 9 team members and an average experience ranging from of 6 to 10 years.

Typically, an agile team comprises of 3 to 4 developers, 1 tester, 1 technical lead, 1 product owner and 1 scrum master



Sprint

The heart of Scrum is a Sprint, a time-box of two weeks or one month during which a potentially releasable product increment is created. A new Sprint starts immediately after the conclusion of the previous Sprint. Sprints consist of the Sprint planning, daily scrums, the development work, the Sprint review, and the Sprint retrospective.

- In Sprint planning, the work to be performed in the Sprint is planned collaboratively by the Scrum Team.
- The Daily Scrum Meeting is a 15-minute time-boxed event for the Scrum Team to synchronize the activities and create a plan for that day.
- A Sprint Review is held at the end of the Sprint to inspect the Increment and make changes to the Product Backlog, if needed.
- The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. In this meeting, the Scrum Team is to inspect itself and create a plan for improvements to be enacted during the subsequent Sprint.

The vital events of scrum are-

- The Sprint
- Sprint Planning
- Daily Scrum Meetings
- The Sprint Review
- The Sprint Retrospective

The Sprint

During a Sprint, a working product Increment is developed. It is usually of duration two weeks or one month, and this duration remains constant for all the sprints in the project. We cannot have varying durations for the different sprints in a project. A new Sprint starts immediately after the conclusion of the previous Sprint.

The Sprint Goal is an objective set for the Sprint. It provides guidance to the Team on why it is building the Increment. It is created during the Sprint Planning meeting. The scope of the sprint is clarified and re-negotiated between the Product Owner and the Team as more about the requirements is learned. Thus, each Sprint is associated with it, a definition of what is to be built, a design, and the flexible plan that will guide building it, the development work, and the resultant product increment.

A Sprint should be cancelled if the Sprint Goal becomes obsolete. This might occur if the organization changes direction or if market or technology conditions change. A sprint can be cancelled only by product owner, though others have an influence on the same.

Due to the short duration nature of Sprints, cancellation during a sprint rarely makes sense. As the sprint cancellations consume resources, for getting re-organized into another Sprint, they are very uncommon.

If a Sprint is cancelled, and part of the work produced during the sprint is potentially releasable, the Product Owner typically accepts it. All the incomplete Sprint Backlog Items are put back into the Product Backlog.

Sprint Planning

The work to be performed in the Sprint is planned in the Sprint Planning Meeting. Sprint Planning Meeting is of duration of maximum of four hours for two weeks sprints and eight hours for one month Sprints. It is the responsibility of the Scrum Master to ensure that the meeting takes place and that all the required attendees are

present and understand the purpose of the scheduled meeting. The Scrum Master moderates the meeting to monitor the sustenance of discussion and closure on time.

Sprint Planning focuses on the following two questions -

- What needs to be and can be delivered in the Sprint Increment?
- How will the work needed for the execution of Sprint be achieved?

The inputs to this meeting are -

- The Product Backlog
- The latest product Increment
- Projected capacity of the Team during the Sprint
- Past performance of the Team

The Scrum Team discusses the functionality that can be developed during the Sprint. Product Owner provides clarifications on the Product Backlog items. The team selects the items from the Product Backlog for the Sprint, as they are the best to assess what they can accomplish in the Sprint. The Team comprises of analysts, designers, developers, and testers. The work is carried out in a collaborative fashion, thus minimizing re-work.

The Scrum Team then comes up with Sprint Goal. The Sprint Goal is an objective that provides guidance to the Team on why it is building the Product Increment. The Team then decides how it will build the selected functionality into a working product Increment during the Sprint. The Product Backlog items selected for this Sprint plus the plan for delivering them is called the Sprint Backlog.

Work during a sprint is estimated during sprint planning and may be of varying size and/or effort. By the end of the Sprint Planning meeting, the work is divided into tasks of duration of one day or less. This is to enable the ease of work allocation, and tracking the completion. If the Team realizes that it has too much or too little work, it can renegotiate the selected Product Backlog items with the Product Owner.

The Team may also invite others (not part of Scrum Team) to attend the Sprint Planning meeting to obtain technical or domain advice or help in estimation.

Daily Scrum Meetings

The Daily Scrum Meeting is a 15-minute meeting for the Team, conducted daily to quickly understand the work since the last Daily Scrum Meeting and create a plan for the next 24 hours. This meeting is also referred to as Daily Stand up Meeting.

The Daily Scrum Meeting is held at the same time and same place every day to reduce complexity.

During the meeting, each Team member explains -

- What did he do yesterday that helped the Team meet the Sprint Goal?
- What will he do today to help the Team meet the Sprint Goal?
- Does he see any impediments that prevent him or the Team from meeting the Sprint Goal?

Daily Scrum is mistaken to be a status tracking event, though, in fact, it is a planning event.

The input to the meeting should be how the team is doing toward meeting the Sprint Goal, and the output should be a new or revised plan that optimizes the team's efforts in meeting the Sprint Goal.

Though the Scrum Master coordinates the Daily Scrum Meeting and ensures that the objectives of the meeting are met, the Meeting is the responsibility of the Team.

If necessary, the Team may meet immediately after the Daily Scrum Meeting, for any detailed discussions, or to re-plan the rest of the Sprint's work.

Following are the benefits of Daily Scrum Meetings -

- Improve communication within the Team.
- Identify impediments, if any, in order to facilitate an early removal of the same, so as to minimize impact on the Sprint.
- Highlight and promote quick decision-making.
- Improve the Team's level of knowledge.

Sprint Review

A Sprint Review is held at the end of every Sprint. During the Sprint Review, a presentation of the increment that is getting released is reviewed. In this meeting, the Scrum Team and the stakeholders collaborate to understand what was done in the Sprint. Based on that, and any changes to the Product Backlog during the Sprint, the attendees arrive at the next steps required that could optimize value. Thus, the objective of Sprint Review is to obtain feedback and progress unitedly.

The Sprint Review is normally held for two hours for two week sprints and for four hours for one month sprints.

The Scrum Master ensures that -

- The meeting takes place.
- The participants understand the purpose.
- The meeting is focused on the required agenda and is completed within the required duration.

The Sprint Review includes the following aspects -

- Attendees include the Scrum Team and key stakeholders, as invited by the Product Owner.
- The Product Owner explains what Product Backlog items have been completed during the sprint and what has not been completed.
- The Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved.
- The Team demonstrates the work that it has completed and answers questions, if any, about the Increment.
- The entire group then discusses on what to do next. Thus, the Sprint Review provides valuable input to Sprint Planning of the subsequent Sprint.
- The Scrum Team then reviews the timeline, budget, potential capabilities, and marketplace for the next anticipated release of the product increment.
- The outcome of the Sprint Review is an updated Product Backlog, which defines the probable Product Backlog items for the next Sprint.

Sprint Retrospective

The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. This is usually a one hour meeting for two-week duration sprints and a three hour meeting for one month duration Sprints.

The purpose of the Sprint Retrospective is to -

- Combine the learnings from the last Sprint, with regards to people, relationships, process, and tools.
- Identify the major items that went well and potential improvements.
- Creation of a plan for implementing improvements to increase product quality.

The Sprint Retrospective is an opportunity for the Scrum Team to introspect and improve within the Scrum process framework so as to make the next Sprint outcome more effective.

Scrum Artifacts provide key information that the Scrum Team and the stakeholders need to be aware of for understanding the product under development, the activities done, and the activities being planned in the project. The following artifacts are defined in Scrum Process Framework -

- Product Backlog
- Sprint Backlog

- Burn-Down Chart
- Increment

These are the minimum required artifacts in a scrum project and project artifacts are not limited by these.

Product Backlog

The Product Backlog is an ordered list of features that are needed as part of the end product and it is the single source of requirements for any changes to be made to the product.

The Product Backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases. Product Backlog items have the attributes of a description, order, estimate, and value. These items are normally termed as User Stories. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.

A Product Backlog is an evolving artifact. The earliest version of it may contain only the initially known and best understood requirements. The Product Backlog gets developed as the product, and the environment in which it will be used, progress. The Product Backlog constantly changes to incorporate what is required to make it effective. As long as a product exists, its Product Backlog also exists.

As the product being built is used and gains value, the Product Backlog becomes a larger and more exhaustive list. Changes in business requirements, market conditions, or technology, cause changes in the Product Backlog, making it a live artifact.

Product Backlog refinement means adding detail, estimates, and priority order to the Product Backlog items. This is an ongoing process performed by the Product Owner and the Team. The Scrum Team decides how and when refinement is to be done.

Product Backlog items can be updated at any time by the Product Owner or at the Product Owner's discretion.

Higher-ordered Product Backlog items are usually clearer and more detailed than lower-ordered ones. More precise estimates are made based on the greater clarity and increased detail. The lower the order, the lesser is the detail.

Product Backlog items that may likely be the candidate requirements for the upcoming Sprint are refined so that these items can be developed during the Sprint. Product Backlog items that can be developed by the Team within one Sprint are deemed to be ready for selection in a Sprint planning meeting.

Sprint Backlog

The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal.

The Sprint Backlog is a forecast by the Team about what functionality will be made available in the next Increment and the work needed to deliver that functionality as a working product Increment.

The Sprint Backlog is a plan with enough detail that can be understood but the Team to track in the Daily Scrum. The Team modifies the Sprint Backlog throughout the Sprint, and the Sprint Backlog emerges during the Sprint. This emergence occurs as the Team works through the plan and learns more about the work needed to achieve the Sprint Goal.

As new work is required, the Team adds it to the Sprint Backlog. As work is performed or completed, the estimated remaining work is updated. When elements of the plan are deemed unnecessary, they are removed. Only the Team can change its Sprint Backlog during a Sprint. The Sprint Backlog is a highly visible, real-time picture of the work that the Team plans to accomplish during the Sprint, and it belongs solely to the Team.

Increment

The Increment is the sum of all the Product Backlog items completed during a Sprint combined with the increments of all previous Sprints. At the end of a Sprint, the new Increment must be a working product, which means it must be in a useable condition. It must be in working condition regardless of whether the Product Owner decides to actually release it.

The Scrum Team needs to have consensus on what is considered to be an Increment. This varies significantly per Scrum Team, but, team members must have a shared understanding of what it means for work to be complete. This is used to assess when work is complete on the product Increment.

The same understanding guides the Team in knowing how many Product Backlog items it can select during a Sprint Planning. The purpose of each Sprint is to deliver Increments of potentially releasable functionality.

Teams deliver an Increment of product functionality every Sprint. This Increment is useable, so a Product Owner may choose to release it immediately. If the understanding of an increment is part of the conventions, standards, or guidelines of the development organization, all Scrum Teams must follow it as a minimum. If it is not a convention of the development organization, the Scrum Team must define a definition of Increment appropriate for the product.

Each Increment is additive to all prior Increments and thoroughly tested, ensuring that all Increments work together.

As Scrum Teams mature, it is expected that their definitions of Increments expands to include more stringent criteria for higher quality. Any one product should have a definition of Increment that is a standard for any work done on it.

Sprint Burn-Down Chart

At any point in time in a Sprint, the total work remaining in the Sprint Backlog can be summed. The Team tracks this total work remaining for every Daily Scrum to project the likelihood of achieving the Sprint Goal. By tracking the remaining work throughout the Sprint, the Team can manage its progress.

Sprint Burn-Down Chart is a practice for trending the work expended by the Scrum Team. This has been proven to be a useful technique in monitoring the Sprint progress towards the Sprint Goal.

The Product Owner tracks this total work remaining at least every Sprint Review. The Product Owner compares this amount with work remaining at previous Sprint Reviews to assess progress toward completing the projected work by the desired time for the goal. This information is shared with all stakeholders.

What is agile project management?

Agile project management is an iterative approach to managing software development projects that focuses on continuous releases and incorporating customer feedback with every iteration.

Software teams that embrace agile project management methodologies increase their development speed, expand collaboration, and foster the ability to better respond to market trends.

Agile Workflow

An agile workflow is a series of stages agile teams use to develop an application, from ideation to completion.

Every software team has a process they use to complete work. Normalizing that process–i.e., establishing it as a workflow–makes it clearly structured and repeatable, which, in turn, makes it scalable. At Atlassian, we take an iterative approach to

workflow management because it helps us meet our goals faster and exemplifies our team culture.

When implementing a workflow for the team, always start simple. Fight the temptation to spend weeks (over-)engineering it. Overly complex workflows are hard to understand and adopt–not to mention adapt. For software teams, we recommend these basic workflow states:

TO DO

Work that has not been started
IN PROGRESS

Work that is actively being looked at by the team
CODE REVIEW

Work that is completed and awaiting review
DONE

Work that is completely finished and meets the team's definition of done
In an issue tracker, these statuses flow from one to the next using transitions that structure the workflow.

Some software teams include additional states in their workflow that help them track the status of work more precisely.

AWAITING QA

Work that has been implemented, but is still waiting for a tester review (see our article on agile testing for more details).
READY TO MERGE

Code that has been reviewed and is ready to merge into the main or release branch.
Each state in the workflow doesn't need to be handled by a different person. As an agile team matures, developers handle more and more of the work–from design all the way through to delivery. An autonomous team that can handle heterogeneous work is one of the hallmarks of agility, after all.

Structuring work in agile

Epic

An agile epic is a body of work that can be broken down into specific tasks (called user stories) based on the needs/requests of customers or end-users. Epics are an important practice for agile and DevOps teams.
When adopting agile and DevOps, an epic serves to manage tasks. It's a defined body of work that is segmented into specific tasks (called "stories," or "user stories") based on the needs/requests of customers or end-users.

Epics are a helpful way to organize your work and to create a hierarchy. The idea is to break work down into shippable pieces so that large projects can actually get done and you can continue to ship value to your customers on a regular basis. Epics help teams break their work down, while continuing to work towards a bigger goal.

Agile epic example

Let's say it's 2050 and we work for a recreational space-travel organization. We do about a dozen launches a year, so each launch isn't the single biggest thing we do in a year, but it's still far from routine and will take many person-hours to complete. That sizing is just right for an epic.

An example epic, "March 2050 Space Tourism Launch" includes stories for routine work items as well as stories aimed to improve key aspects of the shuttle launch, from customers buying space travel tickets to the launch of the rocket itself. As such, multiple teams will contribute to this epic by working on a wide range of stories.

The software team supporting the purchasing of tickets for the March 2050 launch might structure their epic as so:

Epic: March 2050 Launch		
Story: Update date range to include March 2050 Launch dates.	Story: Reduce load time for requested flight listings to < 0.45 seconds	Story: Promote Saturn Summer Sale on confirm page for First Class bookings.

Concurrently, the propulsion teams might contribute to the same epic with these stories:

Epic: March 2050 Launch		
Story: Keep fuel tanks PSI > 250 PPM on launch	Story: Reduce overall fuel consumption by 1%.	Story: Hire new propulsion engineer to replace Gary. #garygate2050

User Story

Summary: *A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.*
It's tempting to think that user stories are, simply put, software system requirements. But they're not.

A key component of agile software development is putting people first, and a user story puts end users at the center of the conversation. These stories use non-technical language to provide context for the development team and their efforts. After reading a user story, the team knows why they are building, what they're building, and what value it creates.

User stories are one of the core components of an agile program. They help provide a user-focused framework for daily work — which drives collaboration, creativity, and a better product overall.