Ashirbad Swain

SDLC (Software Development Life Cycle)

What is SDLC?

- Software Development Life Cycle is a systematic process for building software that ensures the quality and correctness of the software built.
- SDLC process aims to produce high-quality software that meets customer expectations.
- The system development should be complete in pre-defined time frame and cost.
- SDLC consists of a detailed plan which explains how to plan, build and maintain specific software.
- Every phase of the SDLC life cycle has its own process and deliverables that feed into the next phase.

Why SDLC?

here, are the prime reasons why SDLC is important for developing a software system,

- It offers a basis for project planning, scheduling and estimating.
- Provides a framework for a standard set of activities and deliverables.
- It is a mechanism for project tracking and control.
- Increases visibility of project planning to all involved stakeholders of the development process.
- Increases and enhanced development speed.
- Improved client relations.
- Helps you to decreases project risk and project management plan overhead.

SDLC Phases

The entire SDLC process divides into the following stages:

1. Requirement Analysis
2. Feasibility Study
3. Design
4. Coding
5. Testing
6. Installation/Deployment
7. Maintenance

Phase 1: Requirement collection & analysis

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry.

Planning for the quality assurance requirements and recognition of the risks is also done at this stage.

This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities and directives which triggered the project.

Requirement gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

Phase 2: Feasibility Study

Once the requirement analysis phase is completed the next step is to define and document software needs. This process conducted with the help of "**Software Requirement Specification**" document also known as "**SRS**" document. It includes everything which should be designed and developed during the project life cycle.

There are mainly 5 types of feasibility checks,

Economic:

can we complete the project within the budget or not?

Legal:

can we handle this project as cyber law and other regulatory framework/compliances?

Operation feasibility

Can we create operations which is expected by the client?

Technical

Need to check whether the current computer system can support the software.

Schedule

Decide that the project can be completed within the given schedule or not.

Phase 3: Design

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are 2 kinds of design documents developed in this phase:

High-Level Design (HLD)

- Brief description and name of each module
- An outline about the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
- Complete architecture diagrams along with technology details

Low-Level Design (LLD)

- Functional logic of the modules
- Databases tables, which include type and size
- Complete detail of the interface
- Addresses all type of dependency issues
- Listing of error messages
- Complete input and outputs for every module

Phase 4: Coding

Once the system design phase is over, the next phase is coding. In this phase, developers start building the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the **longest** phase of the SDLC process.

In this phase, Developers needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

Phase 5: Testing

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable and working according to the business needs of that system.

Phase 6: Installation/Deployment

Once the software testing phase is over and no longer bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issue if any.

Phase 7: Maintenance

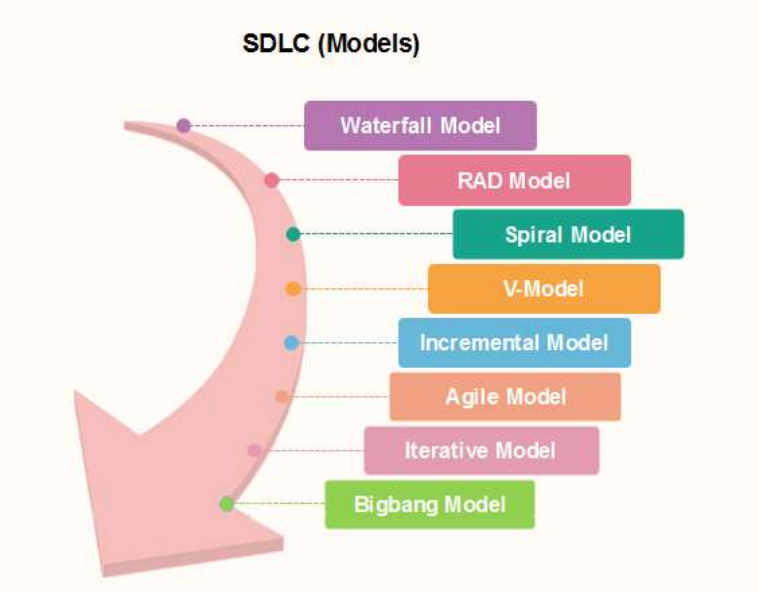
Once the system is deployed and customer starts using the developed system, following 3 activities occur,

- Bug fixing – bugs are reported because of some scenarios which are not tested at all.
- Upgrade – upgrading the application to the newer versions of the software.
- Enhancement – adding some new features into the existing software.

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in first phase.

Popular SDLC models

Here are some most important SDLC models,



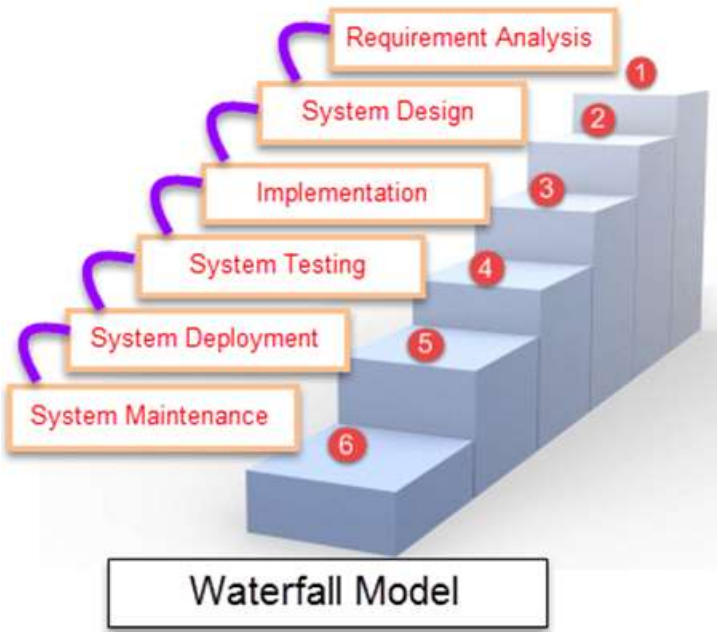
Waterfall Model

Waterfall Model is a sequential model that divide software development into pre-defined phases.

Each phase must be completed before the next phase can begin with no overlap between the phases.

Each phase is designed for performing specific activity during the SDLC phase.

It was introduced in 1970 by Winston Royce and is a universally accepted SDLC model.



Ashirbad Swain

Phases of Waterfall Model

Phase 1: Requirement Analysis

During this phase, detailed requirements of the software system to be develop are gathered from client.

Phase 2: Design

Plan the programming language, for example Java, PHP or .net

Or database like oracle, MySQL etc.

Or the high -level technical details of the project.

Phase 3: Build/Implementation

After design stage, it is built stage, that is nothing but coding the software.

Phase 4: Testing

In this phase, you test the software to verify that it is built as per the specifications given by the client.

Phase 5: Deployment

Deploy the application in the respective environment

Phase 6: Maintenance

Once your system is ready to use, you may later require change the code as per customer request.

When to use SDLC Waterfall Model : Waterfall model can be used when,

- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Requirement is clear
- Environment is stable
- Technology and tools used are not dynamic and is stable
- Resources are available and trained

Advantages

- Before the next phase of development, each phase must be completed
- Suited for smaller projects where requirements are well defined
- They should perform quality assurance test (verification and validation) before completing each stage.
- Project is completely dependent on project team with minimum client intervention.
- Any changes in software is made during the process of the development.

Disadvantages

- Error can be fixed only during the phase
- It is not suitable for complex project where requirement changes frequently.
- Testing period comes quite late in the development process.
- Documentation occupies a lot of time of developers and testers

RAD Model

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.

- Business Modeling
- Data Modeling
- Process Modeling
- Application Generation
- Testing and Turnover

Spiral Model

The spiral model is a **risk-driven process model**. This SDLC model helps the group to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. The spiral technique is a combination of rapid prototyping and concurrency in design and development activities.

Each cycle in the spiral begins with the identification of objectives for that cycle, the different alternatives that are possible for achieving the goals, and the constraints that exist. This is the first quadrant of the cycle (upper-left quadrant).

The next step in the cycle is to evaluate these different alternatives based on the objectives and constraints. The focus of evaluation in this step is based on the risk perception for the project.

The next step is to develop strategies that solve uncertainties and risks. This step may involve activities such as benchmarking, simulation, and prototyping.

V-Model

In this type of SDLC model testing and the development, the step is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

Incremental Model

The incremental model is not a separate model. It is necessarily a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, each cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

Iterative Model

It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.

Big Bang Model

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.

This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements are either unknown or final release date is not given.

Prototype Model

The prototyping model starts with the requirements gathering. The developer and the user meet and define the purpose of the software, identify the needs, etc.

A '**quick design**' is then created. This design focuses on those aspects of the software that will be visible to the user. It then leads to the development of a prototype. The customer then checks the prototype, and any modifications or changes that are needed are made to the prototype.

Looping takes place in this step, and better versions of the prototype are created. These are continuously shown to the user so that any new changes can be updated in the prototype. This process continue until the customer is satisfied with the system. Once a user is satisfied, the prototype is converted to the actual system with all considerations for quality and security.

Agile Model

What is Agile Methodology?

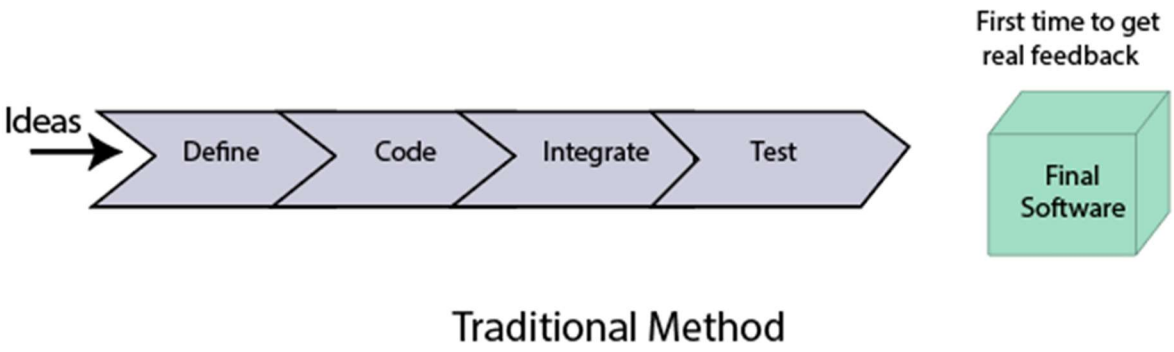
An agile methodology is an **iterative approach** to software development.

Each iteration of agile methodology takes a short time interval of 1 to 4 weeks. So, it distributes the software with faster and fewer changes.

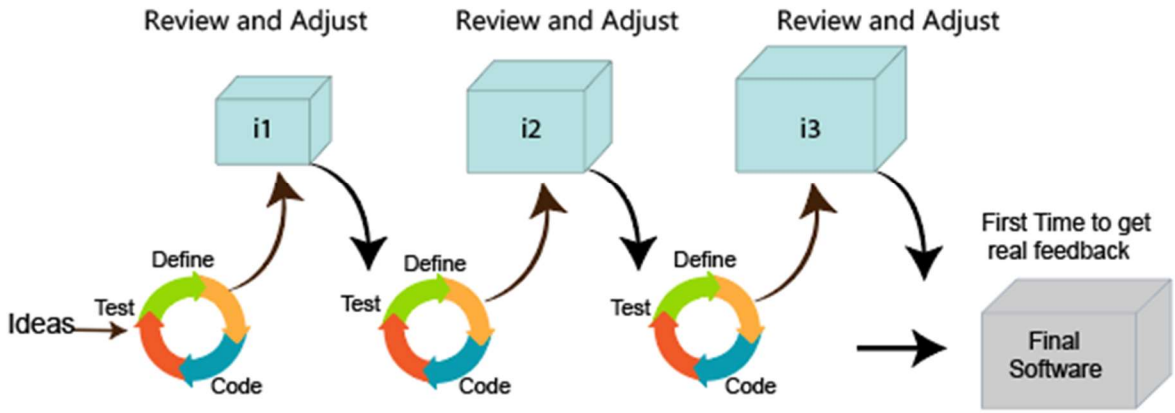
So, the most important endeavour for developing the agile model is to make easy and rapid project achievement.

The agile software development emphasizes on 4 core values,

1. Individual and team interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan



Traditional Method



Agile Method

Agile Methodology

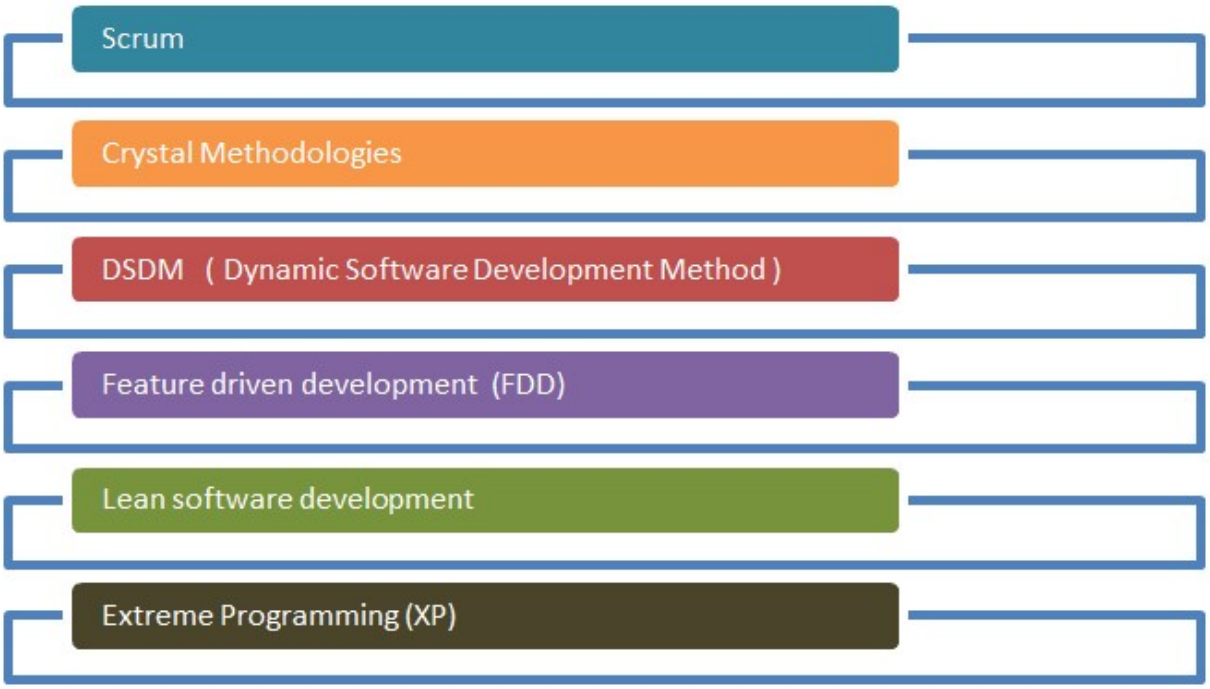
Roles in Agile

There are 2 different roles in an Agile methodology and these are,

1. Scrum Master
2. Product Owner

Also, there are various methods present in agile testing, and these are,

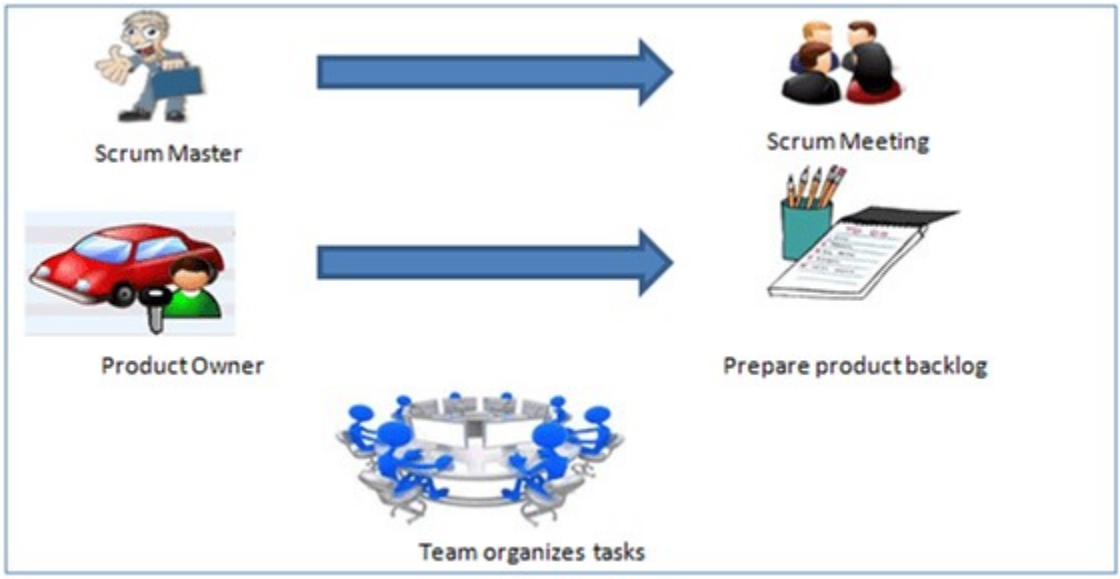
1. Scrum
2. Crystal Methodologies
3. Dynamic Software Development Method (DSDM)
4. Feature Driven Development (FDD)
5. Lean Software Development
6. Extreme Programming (XP)



Scrum

SCRUM is an agile development method which concentrates specifically on how to manage tasks within a team-based development environment.

SCRUM believes in empowering the development team and advocates working in small teams (say 7 to 9 members).



It consists of 3 roles, and their responsibilities are,

Scrum Master

Master is responsible for setting up the team, sprint meetings and removes obstacles to progress.

Product Owner

The product owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration.

Scrum Team

Team manages its own work and organizes the work to complete the sprint or cycle.

Product Backlog

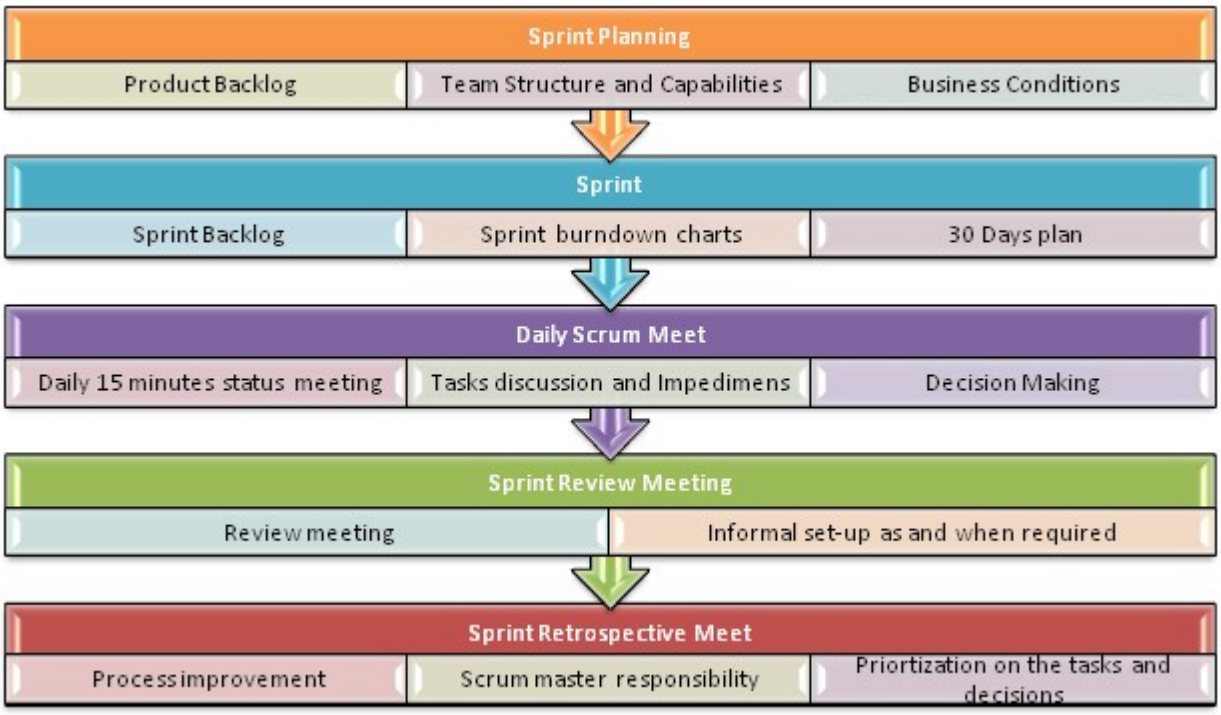
This is a repository where requirements are tracked with details on the no. of requirements (user stories) to be completed for each release.

It should be maintained and prioritized by Product Owner and it should be distributed to scrum team.

Team can also request for a new requirement addition or modification or deletion.

Scrum Practices

Practices are described in detailed:



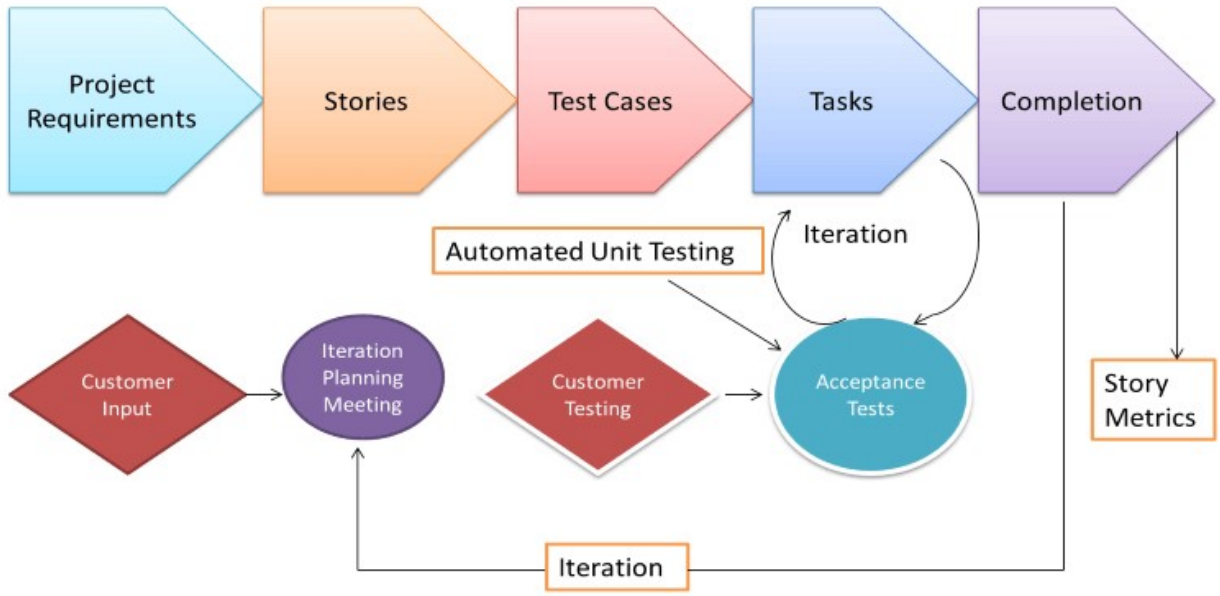
Process flow of Scrum Methodologies

Process flow of scrum testing as follows,

- Each iteration of a scrum is known as Sprint.
- Product backlog is a list where all details are entered to get the end-product.
- During each sprint, top user stories of Product backlog are selected and turned into sprint backlog.
- Team works on the defined backlog
- Team checks for daily work
- At the end of the sprint, team delivers product functionality.

eXtreme Programming (XP)

Extreme Programming technique is very helpful when there is constantly changing demands or requirements from the customers or when they are not sure about the functionality of the system. It advocates frequent "releases" of the product in short development cycles, which inherently improves the productivity of the system and also introduces a checkpoint where any customer requirements can be easily implemented. The XP develops software keeping customer in the target.



Extreme Programming (XP)

Business requirements are gathered in terms of **stories**. All those stories are stored in a place called the **parking lot**.

In this type of methodology, releases are based on the shorter cycles called Iterations with span of 14 days' time period. Each iteration includes phases like coding, unit testing and system testing where at each phase some minor or major functionality will be built in the application.

Phases of extreme Programming

There are 6 phases available in agile XP method,

Planning

- Identification of stakeholders and sponsors
- Infrastructure Requirements
- Security related information and gathering
- Service level agreements and its conditions

Analysis

- Capturing of stories in Parking lot
- Prioritize stories in parking lot
- Scrubbing of stories for estimation
- Define iteration SPAN (time)
- Resource planning for both Development & QA teams

Design

- Break down of tasks
- Test scenario preparation for each task
- Regression automation framework

Execution

- Coding
- Unit testing
- Execution of Manual test scenarios
- Defect report generation
- Conversion of manual to automation regression test cases
- Mid iteration review
- End of iteration review

Wrapping

- Small releases
- Regression testing
- Demos and reviews develop new stories based on the need
- Process improvements based on end of iteration review comments

Closure

- Pilot Launch
- Training
- Production Launch
- SLA Guarantee assurance
- Review SOA strategy
- Product support

There are two storyboards available to track the work on a daily basis, and those are listed below for reference.

- Story Cardboard

  This is a traditional way of collecting all the stories in a board in the form of stick notes to track daily XP activities. As this manual activity involves more effort and time, it is better to switch to an online form.

- Online Storyboard

  Online tool Storyboard can be used to store the stories. Several teams can use it for different purposes.

Crystal Methodologies

Crystal Methodology is based on three concepts

1. Chartering:

   Various activities involved in this phase are creating a development team, performing a preliminary feasibility analysis, developing an initial plan and fine-tuning the development methodology

2. Cyclic delivery:

   The main development phase consists of two or more delivery cycles, during which the

   1. Team updates and refines the release plan
   2. Implements a subset of the requirements through one or more program test integrate iterations
   3. Integrated product is delivered to real users
   4. Review of the project plan and adopted development methodology

3. Wrap Up:

   The activities performed in this phase are deployment into the user environment, post- deployment reviews and reflections are performed.

Dynamic Software Development Method (DSDM)

DSDM is a Rapid Application Development (RAD) approach to software development and provides an agile project delivery framework.

The important aspect of DSDM is that the users are required to be involved actively, and the teams are given the power to make decisions.

Frequent delivery of product becomes the active focus with DSDM.

The techniques used in DSDM are,

1. Time Boxing
2. MoSCoW Rules
3. Prototyping

The DSDM project consists of 7 phases,

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Module Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

Feature Driven Development (FDD)

This method is focused around "**designing & building**" features. Unlike other agile methods, FDD describes very specific and short phases of work that has to be accomplished separately per feature. It includes domain walkthrough, design inspection, promote to build, code inspection and design. FDD develops product keeping following things in the target

1. Domain object Modeling
2. Development by feature
3. Component/ Class Ownership
4. Feature Teams
5. Inspections
6. Configuration Management
7. Regular Builds
8. Visibility of progress and results

Lean Software Development

Lean software development method is based on the principle "**Just in Time Production**". It aims at increasing speed of software development and decreasing cost. Lean development can be summarized in **7** steps,

1. Eliminating Waste
2. Amplifying Learning
3. Defer Commitment
4. Early delivery
5. Empowering the team
6. Building integrity
7. Optimize the whole

Kanban

Kanban originally emerged from Japanese word that means, a card containing all the information needed to be done on the product at each stage along its path to completion. This framework or method is quite adopted in software testing method especially in agile testing.

When to use the Agile model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Advantages

- Frequent delivery
- Face-to-Face communication with clients
- Efficient design and fulfils the business requirement
- Anytime changes are acceptable
- It reduces total development time

Disadvantages

- Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
- Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

Difference Between Agile Methodology & Waterfall Model

Agile Methodology	Waterfall Model
It follows the incremental approach.	It is a sequential design process.
It divides the project development lifecycle into a sprint.	The software development process is divided into distinct phases.
Agile methodology is a flexible methodology.	The waterfall model is a structured software development methodology.
Agile is the collection of many different projects.	It is completed as one single project.
The test plan is reviewed after each sprint.	Test plan is reviewed after complete development.
Testing team can take part in the requirements change phase without problems.	It is difficult for the test to initiate any change in needs.

Agile Manifesto

In February 2001, at the Snowbird resort in Utah, a team of 17 software developers met to discuss lightweight development methods. The result of their meeting was the following Agile Manifesto for software development: -

We are uncovering the better ways of developing software by doing it and helping others to do it. Through this meeting, we have come to value -

- Individuals and interactions over Processes and tools.
- Working software over comprehensive documentation.
- Customers are collaboration over contact negotiation.
- Responding to change over following a plan.

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

The 12 Principles of Agile Manifesto

1. **Customer Satisfaction:** Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.
2. **Welcome Change:** Making changes during software development is common and inevitable. Every changing requirement should be welcome, even in the late development phase. Agile process works to increase the customers' competitive advantage.
3. **Deliver the Working Software:** Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest time period.
4. **Collaboration:** Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.
5. **Motivation:** Projects should be built around motivated team members. Provide such environment that supports individual team members and trust them. It makes them feel responsible for getting the job done thoroughly.
6. **Face-to-face Conversation:** Face-to-face conversation between Scrum Master and development team and between the Scrum Master and customers for the most efficient and effective method of conveying information to and within a development team.
7. **Measure the Progress as per the Working Software:** The working software is the key and primary measure of the progress.
8. **Maintain Constant Pace:** The aim of agile development is sustainable development. All the businesses and users should be able to maintain a constant pace with the project.
9. **Monitoring:** Pay regular attention to technical excellence and good design to maximize agility.
10. **Simplicity:** Keep things simple and use simple terms to measure the work that is not completed.

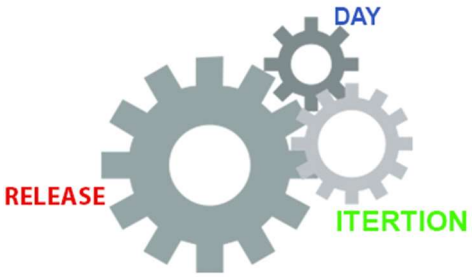
11. **Self-organized Teams:** The Agile team should be self-organized. They should not be depending heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
12. **Review the Work Regularly:** The work should be reviewed at regular intervals, so that the team can reflect on how to become more productive and adjust its behavior accordingly.

Agile Characteristics

The product developed under agile methodology has seen important characteristics that are given below,

Agile Development Releases & Fixed-Length Iterations

- The agile software development method is based on two central units of delivery: **release** & **iteration**.
- A single version consists of several iterations phase. Each iteration consists of its **micro-project**.
- The different functions of agile development like defects, enhancement requests and other work items are organized, estimated, and prioritize and then assigned to release.



Agile Development Delivers-Working, Tested Software

- The primary measure of the agile development team is to deliver working, progress and tested feature software.
- Working feature serve as the basis for enabling and improving customer feedback. It also serves as team collaboration, and overall project visibility. They provide such evidence so that the system and the project are on track.
- At every step of product development, the team continuously works to assemble on the best business solution. This is done using the latest input from users, customers and other stakeholders.

Value – Driven Development

Agile development methodology focuses really on delivering business value early and continuously. It is measured by running tested software. The development team focuses on product features as the central unit of planning, tracking, and delivery.

As the development goes on from iteration to iteration, the team tracks how many products are running, tested features they are delivering.

Continuous (Adaptive) Planning

As the project launches, the development team does just more planning to get going with the initial iteration and, if it is appropriate, to lay out a high-level release plan of features. The single iteration leads the key to continuous planning.

As the iteration starts, the team choose a set of features to implement, determines and estimates each technical task for each feature.

Multi-Level Planning in Agile Development

The continuous planning impacts much more significant result if it occurs on at least two levels:

- At the **release level**, the development team identifies and prioritize the features they must have, would like to have, and they can do within the deadline.
- At the **iteration level**, development team picks and plan for the next batch of features to implement, in priority order. If the product features are too large to estimated or delivered within a single iteration, the development team break them down further.

Relative Estimation

Several agile development teams use the practice of relative estimation for features to accelerate planning. It removes unnecessary complexity. The development team selects a few (3-5) relative estimation categories, or buckets, and estimates all features in terms of these categories.

The concept of relative estimation or/and predefined estimation buckets that prevent the team from wasting time on debating. When the product feature exceeds an agreed maximum estimate, then it should be further broken down into multiple features.

Emergent Feature Discovery

As disputed to spending weeks or months, analyzing the requirements before initiating development, agile development projects quickly prioritized and estimated features, and then refine the details when required. The feature of the product is described in more detail between customers, testers, and developers working together.

Continuous Testing

Using continuous testing of software product, we determine the progress and prevent defects. We handle the running and tested features. Using continuous testing, we can reduce the failure risk in the project.

Continuous Improvement

Continuous testing and constant improvement are correlated with each other. While continuous testing, if we found any bugs or project failure, we continuously improve that bugs immediately. We continuously refine both the project and the system.

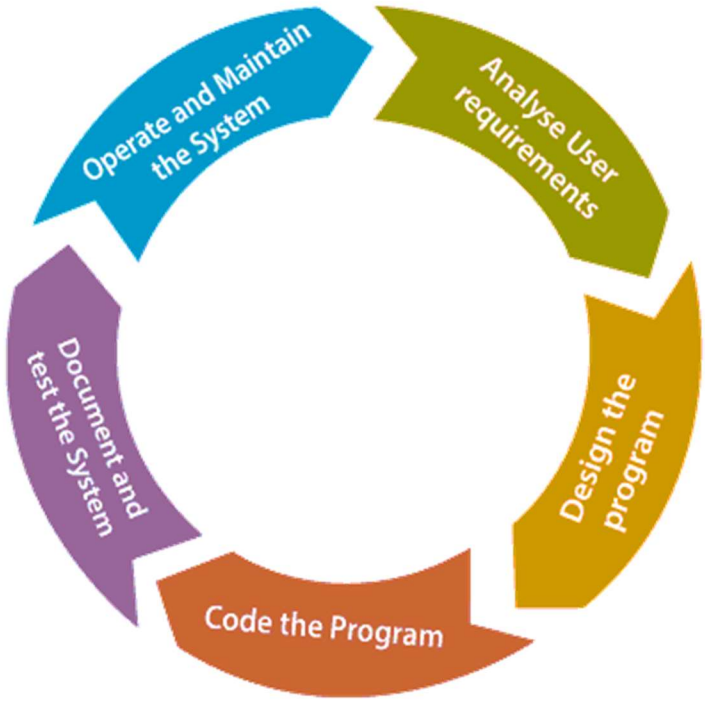
Small, Cross-Functional Teams

The incremental software product is delivered at every iteration. The development teams must also be cross-functional to be successful in developing the valuable software.

Agile Software Development Life Cycle (SDLC)

Agile Software Development Life Cycle (SDLC) is the combination of both **iterative** and **incremental** process models. It focuses on process adaptability and customer satisfaction by rapid delivery of working software product. Agile SDLC breaks down the product into small incremental builds. These builds are provided into iterations.

In the agile SDLC development process, the customer is able to see the result and understand whether he/she is satisfied with it or not. This is one of the advantages of the agile SDLC model. One of its disadvantages is the absence of defined requirements so, it is difficult to estimate the resources and development cost.



Each iteration of agile SDLC consists of cross-functional teams working on various phases:

1. Requirement gathering and analysis
2. Design the requirements
3. Construction/ iteration
4. Deployment
5. Testing
6. Feedback

Ashirbad Swain

Requirements gathering and analysis

In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

Design the requirements

When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

Construction/ Iteration

When the team defines the requirements, the work begins. The designers and developers start working on their project. The aims of designers and developers deploy the working product within the estimated time. The product will go into various stages of improvement, so it includes simple, minimal functionality.

Deployment

In this phase, the team issues a product for the user's work environment.

Testing

In this phase, the Quality Assurance team examine the product's performance and look for the bug.

Feedback

After releasing of the product, the last step is to feedback it. In this step, the team receives feedback about the product and works through the feedback.

Agile SDLC Process Flow

1. Concept: Project are imagined and prioritized.
2. Inception: Team members are created, funding is put in place, and basic environments and requirements are discussed.
3. Iteration/Construction: The software development team works to deliver working software. It is based on requirement and feedback.
4. Release: Perform quality assurance (QA) testing, provides internal and external training, documentation development, and final version of iteration into the product.
5. Production: It is ongoing support of the software.

Advantages of Agile SDLC

1. Project is divided into short and transparent iterations.
2. It has a flexible change process.
3. It minimizes the risk of software development.
4. Quick release of the first product version.
5. The correctness of functional requirement is implemented into the development process.
6. Customer can see the result and understand whether he/she is satisfied with it or not.

Disadvantages of Agile SDLC

1. The development team should be highly professional and client-oriented.
2. New requirement may be a conflict with the existing architecture.
3. With further correction and change, there may be chances that the project will cross the expected time.
4. There may be difficult to estimate the final coast of the project due to constant iteration.
5. A defined requirement is absent.

Agile Project Management

Agile project management is an interactive approach to manage software development. The agile project management focuses on continuous releases and covers customer feedback with every iteration.

Traditionally the agile project management is classified into two frameworks: **scrum** and **Kanban**. The Scrum framework focused fixed-length project iterations, whereas Kanban framework focused on continuous releases. After competition of project first iteration (or steps) project management activity immediately moves on to the next.

History of Agile Project Management

Agile project management is rapidly rising in the 21st century. It is used for software development projects and other IT initiatives.

However, from the mid-20th century, the concept of continuous development has taken various forms. For example, there was James Martin's **Rapid Iterative Production Prototyping (RIPP)**, an approach that served as the premise for the 1991 book **Rapid Application Development (RAD)**.

The agile project management framework which has emerged in most recent years is known as Scrum. This methodology features works on the development team to create a product backlog. It also creates a prioritized list of the features, functionalities, and fixes required to deliver a successful software system. The scrum team offers the pieces of a task in rapid increments.

How Agile Project Management works

The agile project management calls for teams to regularly evaluate cost and time as they move through their work. They use velocity, burnup and burndown charts to measure their work, rather than Gantt charts and project milestones to track progress.

The agile team practices to continuous development and continuous integration using technology that automates steps to speed up the release and use of products.

The presence and participation of the project manager are not required in agile project management. Although the presence of the project manager is essential for success under the traditional (waterfall model) project delivery. The role of the project manager is to distribute task among team members. However, the project manager is not obsolete in agile project management, and many organizations use them in a large, more complex project. The organization mostly places them in the project coordinator role.

Scrum

What is Scrum?

Scrum is a framework that helps agile teams to work together. Using it, the team members can deliver and sustain the complex product. It encourages the team to learn through practice, self-organize while working on the problem. Scum is a work done through the framework and continuously shipping values to customers.

It is the most frequent software that is used by the development team. Its principle and lessons can be applied to all kinds of teamwork. Its policy and experiences are a reason of popularity of Scrum framework. The Scrum describes a set of tools, meetings, and roles that help the team's structure. It also manages the work done by the team

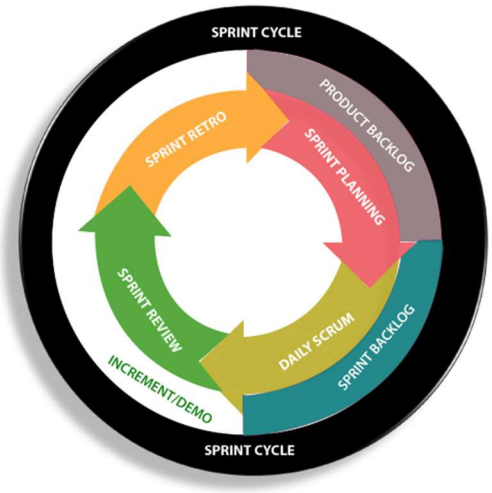
The Framework

Scrum and agile are not the same thing because Scrum focused on continuous improvement, which is a core foundation of agile. Scrum framework focuses on ongoing getting work done.

What are sprints?

With scrum, a product is built in a series of repetition called **sprints**. It breaks down big complex projects into bite-size pieces. It makes projects more manageable, allows teams to ship high quality, work faster, and more frequently. The sprints give them more flexibility to adapt to the changes.

Sprints are a short, time-boxed period for Scrum team that works to complete a set amount of work. Sprints are the core component of Scrum and agile methodology. The right sprints will help our agile team to ship better software.



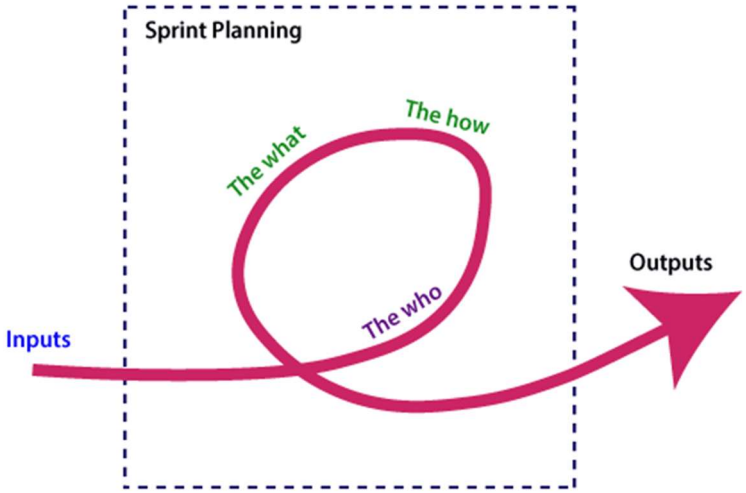
What is sprint plan?

Sprint plan is an action in Scrum that kicks off the sprint. The primary purpose of sprint plan is to define what can deliver in the sprint. It also focuses on how the work will be achieved. It is done in combination with the whole Scrum team members.

The sprint is a set of the period where all the work to be done. Before we start the development, we have to set up the sprint. We need to describe how long time is required to achieve the sprint goal and where we are going to start.

Factors affecting Sprint planning

- **The What:** The product owner describes the goal of the sprint and the backlog items which contribute to achieve that goal.
- **The How:** Agile development team plans its necessary work on how to achieve and deliver the sprint goal.
- **The Who:** The product owner defines the goal based on the value that the customers seek. And the developer needs to understand how they can or cannot deliver that goal.
- **The Inputs:** The product backlog provides the list of input stuff that could potentially be part of the current sprint. The team looks over the existing work done in incremental ways.
- **The Outputs:** The critical outcome of sprint planning is to meet described team goal. The product set the goal of sprint and how they will start working towards the goal.



What is the product backlog?

A product backlog is a registered list of work for the development team. It is driven from the roadmap and its requirements. The essential task is represented at the top of the product backlog so that the team member knows what to deliver first. The developer team doesn't work through the backlog from the product owner's side and product owner doesn't push the work to the developer team. The developer team pulls work from the product backlog.

Backlog starts with the two "R"s

The fundamental product backlog is provided by a team's **roadmap** and **requirements**. Roadmap repetition breaks down into several epics, and each epic will have several requirements and user stories.

The product owner organizes each of the customer stories into a single list. This story is organized for the development team. The product owner chooses to deliver first complete epic.

The factors that influence a product owner's prioritization

- Priority of customer
- Importance of getting feedback
- Relative implementation difficulty
- Symbiotic relationships between work items

Kanban

What is Kanban?

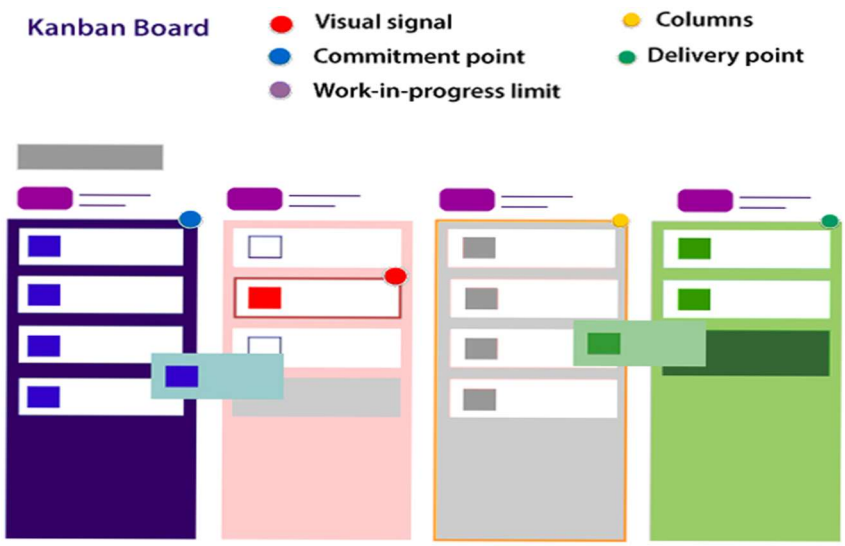
Kanban is a popular framework which is used to implement agile software development. It takes real time communication of capacity and complete transparency of work. The work items are represented in a Kanban board visually, allowing team members to see the state of every piece of work at any time.

Boards

The Kanban board is the agile project management tool that designed the necessary visualized work, limited work-in-progress, and maximizes flow (or efficiency). It uses cards, columns, and provides continuous improvement to help technology and service teams who commit the right amount of work and get it done.

Elements of a Kanban board

A person called David Anderson divides the Kanban board into five different components. These are Visual signals, columns, work-in-progress limits, a commitment point, and a delivery point.



1. **Visual Signals:** The Kanban board is a visual card (stickies, tickets, or otherwise). Kanban team write their projects and work items onto cards, usually per person each card. For agile teams, each card could encapsulate into one user story. Once the board completed, this visual team helps team members and stock members quickly to understand what the team is working.
2. **Columns:** The column represents the specific activities that compose a "workflow" together. The card flows through a workflow until its completion. The workflow may be a simple as "To Do," "In Progress," "Complete," or much more complicated.
3. **Work in progress (WIP) Limits:** The work in progress limits are the maximum number of cards which can be in one column. This is at any given time. It gives the alert signal that you committed too much work.

4. **Commitment point:** Kanban teams also maintain a backlog for their board. This is where the customers and team member put ideas for projects that the team can pick up. The team members pick up plans when they are ready. The committed point is a movement where the design is picked up by the team, and work starts on the project.
5. **Delivery point:** It is the end point of a Kanban team's workflow. Mostly the delivery point for every team is when the product and services are handed to the customer.

Kanban Vs Scrum Board

Kanban	Scrum
Kanban is an ongoing process.	Scrum sprints have a start and stop dates
Kanban has no formal roles.	Role is clearly defined of each team in the scrum (product owner, development team, and scrum master). Both teams are self-organized.
A Kanban board is used throughout the lifecycle of a project	Scrum board is cleared and recycled after each sprint.
This board is more flexible with regards to tasks and timing. Its task can be reprioritized, reassigned, or updated as needed.	This board has the number of tasks and a strict deadline to complete them.

Definition of Kanban

Kanban is the combined methodology for implementing agile software development. Like scrumban, it is a combination of both scrum and Kanban (mixed) methodology.

Difference between Agile & Scrum

Agile is an iterative approach of software development methodology using short iterations of 1 to 4 weeks. Due to the agile methodology, the development process is aligned to deliver the changing business requirement. Using Agile methodology, the software is distributed with faster and fewer changes.

Scrum is a framework of agile that helps agile teams to work together. Using it, the team members development, deliver and sustain the complex product. It encourages the team to learn through practice, self-organize while working on the problem. Scum is a work done through the framework and continuously shipping values to customers.

Agile

1. Agile is an **iterative and incremental approach** to software development methodology.
2. In this approach, the **leadership** plays an important role.
3. Agile software development is highly suitable for the **medium or large project**.
4. **Flexibility** is the most significant advantage of agile as it quickly reacts to changes.
5. Agile involves **face-to-face communication** and collaboration between the members of various cross-functional teams.
6. Agile development needs **frequent delivery** to the end user for their feedback.
7. In this development, each step like requirements, analysis, design, are **continually monitored** during the lifecycle.
8. The **project leader** takes cares of all the tasks in the agile method.
9. End-user may give their **feedback during the development** process. So, the end product will be more useful.
10. **Delivery and update** of the software are taking place regularly.
11. Design and execution should be kept **simple**.
12. The priority of agile development is always to satisfy the customer by providing **continuous delivery** of valuable software.
13. Working software is the most **fundamental measure** of progress.
14. It is best to have **face-to-face communication** to get as close to the project goal as possible.

Scrum

1. Scrum is a framework of agile methodology. In which **incremental builds** are delivered to end user in every two to three weeks.
2. Scrum's team is **self-organized**, cross-functional team.
3. Scrum is used in the project where the requirement **rapidly** changes. v
4. A compared to agile it is more **rigid**. So that there are no chances of frequent change.
5. In **daily stand up meeting** the teamwork is achieved with a fixed role assigned to team members, scrum master, and product owner.
6. **No need to change many more** while implementing scrum process.
7. In this process, a **build is delivered** after each sprint to the client for their feedback.
8. After every sprint a demonstration of functionality is provided. So that the **regular feedback** can be taken before next sprint.
9. There is no team leader, so the **entire team handles the issues** or problems.
10. When the team completes the **current sprint activity**, then the next sprint is planned.
11. Design and execution can be **innovative and experimental**.
12. The **daily sprint meeting** is organized to review the feedback to decide the future progress of the project.
13. Working software is **not a fundamental measure**.
14. The target of the Scrum team is to deliver **maximum business value**.

Agile Daily Stand-up

Agile daily stand-up is termed as the day-to-day status meeting on the project of the members of the agile team. The daily meeting of the agile team discussed the forum for regular updates as well as the problems of team members. It focuses on addressing the issues and tries to solve the issues quickly. The daily stand-up is the regular practice, no matter how an agile team is established regardless of its office location.

What is Daily Stand-up?

The daily stand-up is a daily status meeting of the agile team member. This meeting roughly takes 12 to 18 minutes (an average of 15 minutes).

Each member of the team has to answer three important questions

1. What he/she did yesterday?
2. What he/she will do today?
3. The problem he/she is facing . . . He/she blocked due to. . .

The daily stand-up is done for a day-to-day status update. The meeting of team members with the product owner can be scheduled at different time. The participants in the stand-up meetings only stand instead of sitting so that the meetings get finished quickly.

Ashirbad Swain

Important of Stand-up:

The importance of having a daily stand-up in agile are as follows:

- The team can evaluate the progress report daily.
- The team member discusses all the progress and the commitments he/she made for the day.
- The members can also see whether they can deliver the project as per the iteration plan or not.
- Stand-up provides visibility to the team on any delay that occurs due to some obstacles.

Who Attends a Stand-up?

- The project owner, scrum master, and the delivery team should attend the stand-up regularly.
- Customers and Stakeholders are encouraged to participate in the meeting, and they act as an observer. However, they are not supposed to participate in stand-ups.
- The responsibility of scrum master is to take note each team member's queries and the problems they are facing.

Geographically Dispersed Teams

A stand-up meeting is done in different ways depends upon the working time zone.

- On the rotation basis, select a member who can take the stand-up meeting of teams located in different time zones.
- A separate team has a separate stand-up meeting.
- Daily update the status of the stand-up in a tool such as SharePoint, Rally, Wikis, etc.
- There are varieties of communication tools ready like video conferencing, instant messengers, conference call, and other knowledge sharing tools.

Agile Definition of Done

Agile Definition of **done** is defined into three different stages called User Story (Requirement), Iteration, and product Release. These are given below:

User Story (requirement)

A user story is a requirement which is formulated into few sentences. The user requirement is the everyday language of user. This user story should be completed within iteration. The user story is done when

- All the related code and documentation have been checked-in.
- The product passed all the processes of unit test.
- All the processes of the acceptance test case have been moved.
- The product owner must have accepted the story.
- The help text (documentation) is written.

Iteration

An iteration is a time-based collection of a user story. It works on the defected product and accepted within the release of a product. Iteration is defined at the time of the iteration planning meeting and completed within the iteration demo and review meeting. The iteration is also known as a sprint. The repetition is required when:

- Performance of the product has been tested.
- Product backup is complete.
- User requirement has been accepted or moved for the next iteration.
- Defected product has been fixed or postponed for the next iteration.

Release

The product release is a major occasion that represents an internal and external delivery of work. It also tests the version of the product or system. The product release is done when:

- The system is stress tested.
- Performance is high.
- Contain the security validation in the product.
- Disaster recovery plan is tested.

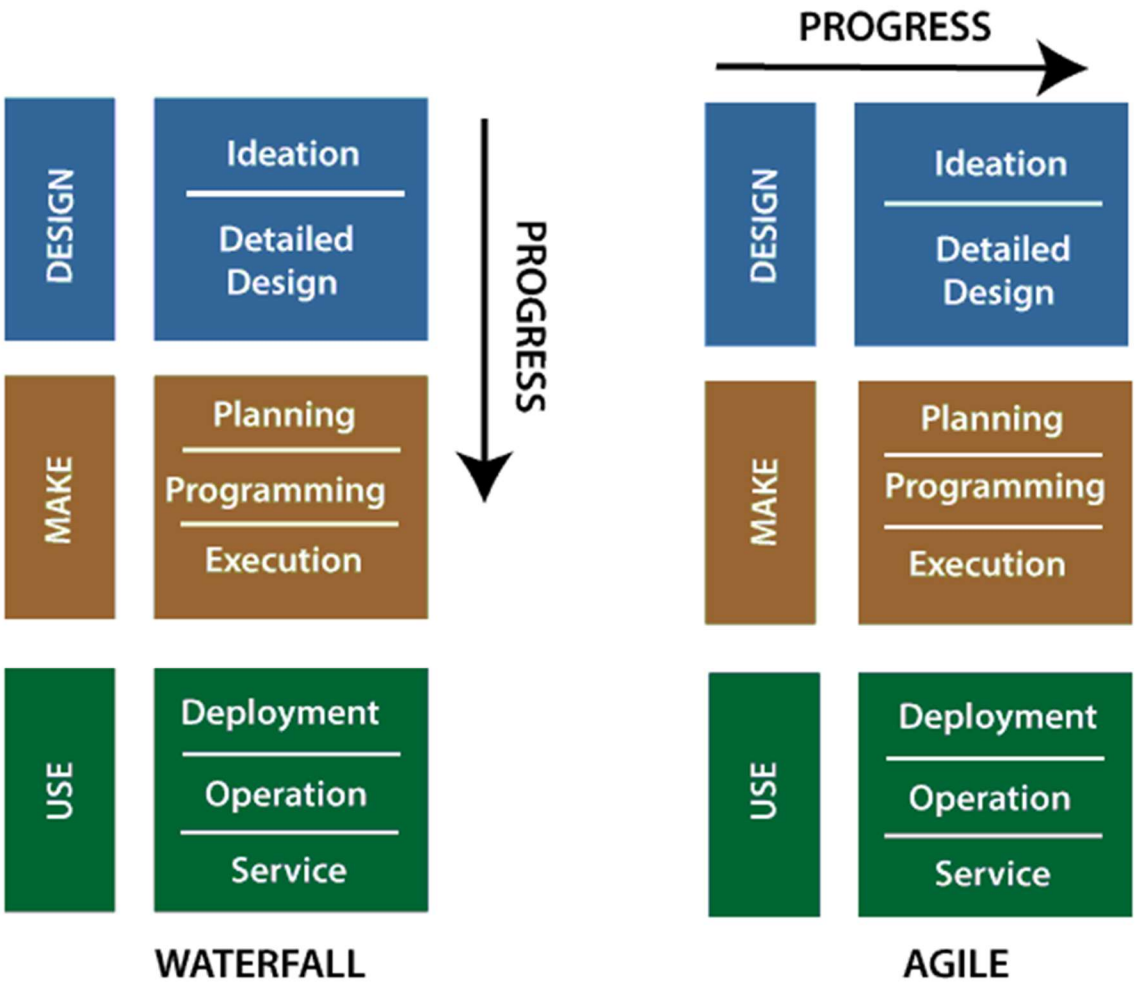
Agile Design

Design plays a vital role in any software development process. The agile team also focuses on "what to do about design" because of the following four factors:

- Many crucial factors focus on loyal designs during the planning process. Design forces towards waterfall culture throughout product implementation.
- Designers also interact with a cross team for a limited time.
- Designers don't always have an easy way to report feedback to the engineering team.
- The presentation and logic layers are not still transparent. They are not separated clearly in the code base, making style changes difficult.

The product design process and customer interview

The agile is divided into several methodologies and processes. These methodologies and processes keep the iterative and free-flowing nature of the technique at their core. The agile design and development methodology used especially in engineering development, and this process called Scrum.



Ashirbad Swain

Customer interviews can be an informative part of the project design phase. We will have several of those "light bulb" movements during interviews. It encourages the people who are interviewing with other members of the team (engineering, marketing, design, etc.)

There are several resources that are available on which we conduct an interview- the logistics, methods, and techniques.

The customer interview pyramid:

Atlassian is a simple framework that helps in building the customer interview pyramid. This pyramid looks like as

Customer interview Pyramid

Communication Observation: At the bottom of the pyramid, we will get the very minimum. We should all come back from an interview and be able to list observations as we don't need any experience to regulate what you've seen.

Interpret problems: Above the Communication Observation, it is an interpret problem. It is explaining the user's behavior and grouping them with an over-arching problem statement.

Connecting opportunities: This is the peak of the pyramid where the most value comes in combining the problem with potential opportunities or related patterns. This helps influence a roadmap and make decisions about what to tackle next.

Ashirbad Swain

Agile Software development

Agile development is more than a framework such as Kanban, Scrum, and Extreme Programming of Feature-Driven Development (FDD). It is more than practice, such as planning, test-driven development, planning sessions, stand-ups, and sprints.

Agile software development contains the set of frameworks, so it is called as an umbrella term. These frameworks are based on the values and principles expressed in Agile manifesto. .

The things that separate agile from other approaches to software development are the focus on the people doing the work and how they work together. The agile software development communities focus on collaboration and the self-organizing team.

Mostly the team and organization start work doing on agile software development, and then they focus on practices that help with collaboration and organizing the work.

How to Be an Awesome Agile Developer

The agile team developers focus on sustainable development? Not heroics. Software sustainability is good estimation, effective branching strategies for managing code. This code is executed by automated testing to protect the quality, and continuous deployment to get fast feedback from users. Agile development is a continuous deployment to get quick feedback from users.

The "**iron triangle**" is a project management system in which all the developers should know about the project scope, schedule, and quality development.

Journey to a stress-free software release:

The success measure of an agile team is that when the working software product is released to the customer. But some time, it is found that the software teams feel the terrible experience at the time of validating the completed issues against artifacts. The code review might be missing. The complete code is not being merged, builds for merged code fail, etc.

Factors that build success software release:

Code best practice: it will improve the ability to deliver a quality product. The code review is essential before providing the product, and monitoring and fixing the declining builds will assure faster time to release.

Set up and maximize Jira Software's release hub: The team focus on setup the Jira software's release hub. It saves the working hours by allowing the release hub to provide a clear picture of progress status and release.

Automation from build code to release: The complete automation from build code to releasing the version straight from release hub.

Why code reviews matter:

Code review is an essential part of the software development before releasing to customer. It helps developers to learn the code base, as well as help them to learn a new technology that grows their skill sets.

What is exactly a code review? When the developer team finishes their working on an issue, other developers pay attention to the code and consider questions like:

- Is there any accessible, logical error in the code?
- Is there any module that takes an outside requirement, and all the cases are fully implemented?
- Is the new automated test sufficient for the new code? Is there any requirement to rewrite in the existing automated tests for changing the code?
- Is the new code conforming to current style guidelines?

Product Management

Product management is the organizational function which guides every step to product lifecycle. The product lifecycle starts from development to positioning and pricing. Its focus on the product and its customer first and foremost.

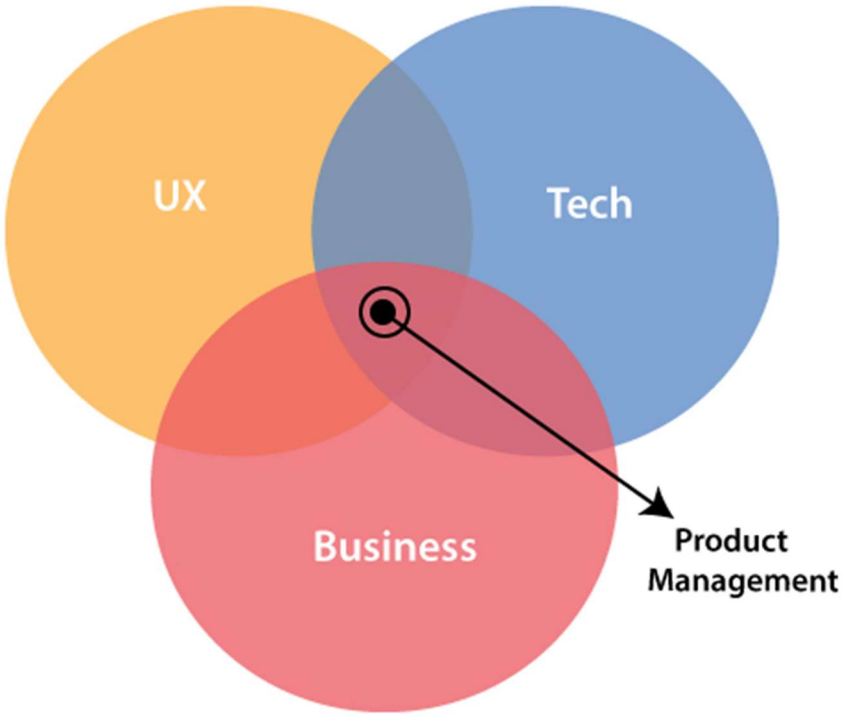
The product entirely focuses on the customer. Product teams routinely ship, better designed, and higher performing product. The product team members daily work with product managers and have interviewed dozens of their role and responsibility. In product management, there is no such one way to apply any principle. Every product has its own goals and challenges which require a unique and customized approach to product management.

Factor affecting product management

Business: Product management helps teams to achieve their business objective by minimizing the communication gap between product developments, design, the customer, and the company.

User Experience: Product management concentrate on the user experience (UX) that represents the customer within an organization. Better UX is focus manifests itself.

Technology: Product management is a day-to-day activity in the engineering department. The accurate understanding of computer science is paramount.



What is Agile Product Management?

Agile product management is about guiding software development, product management through multiple iterations. As agile programs are more fluid than traditional approaches so that agile product management is a more flexible approach.

Agile product managers are more integrated towards technology team than business teams. The product management is supported by the management team and Product Marketing Managers to round out the product discipline. The product manager works over marketing data and business objective.

Product Roadmap

The product roadmap is a shared resource of truth. It is the outline of the vision, priorities, direction, and progress of product over time. The roadmap is the plan of action which aligns the organization around short-term and long-term goals for the product or project. The roadmap also plans how the product and project will be achieved.

The item on the roadmap should be clearly linked to your product strategy. It should also be responsive to changes in customer feedback and the competitive landscape. The product owner uses this roadmap to collaborate with their teams and build consent on how a product will grow and shift over time. This roadmap provides a team to keep everyone on the same page and gain context for their everyday work and future direction.

Tips for presenting Product Roadmaps

The presentation of the roadmap can be a tedious task for both developed and product managers. Several tips used to present the product roadmap are given below:

1. Set the right context
2. Consider commitments carefully
3. Make realistic plans
4. Think big, start small
5. Build a business case
6. Balance mundane with motivating
7. Roll with the punches
8. Be open and honest

Agile Product Requirement

Building a high and new product requires research and comprehensive planning. But a question arises from where to start? The product manager generally begins with the product requirements document (PRD).

A product requirement document defines the product that we are building. Product requirement outlines the product's ambition, its features, functionalities, and behavior.

After collecting all the product requirements, product manager shared it with stakeholders - business and technical teams. They help in building, launch, or market the product.

Agile Scale

In an Agile methodology, there are two popular frameworks- **scrum** and **Kanban**. A team level uses scrum and Kanban as a framework. As their popularity increases, the industries start to scale agile to suit larger organizations. There are two popular methods emerge to facilitate, these are **a scrum of scrums**, and the **Scaled Agile Framework (SAFe)**. The scrum and Kanban are high starting points for scaling agile within an organization.

Scrum of Scrums

It is the most attractive, agile framework for individual teams. When several scrum teams work together on a big project, the scrum of scrums is the next step for scaling agile. The most crucial component of the scrum of scrums is a multi-team stand-up. It is a small meeting for scrum masters to talk about the agile process.

Select a member from each team to get a start, and each team represents them at the scrum of scrums, admirably someone in a technical role. It is a domestic meeting where the scrum master helps to facilitate the stand-up, but it is run just like any other team stand-up.

Scaled Agile Framework (SAFe)

Scaled Agile Framework (SAFe) is another way to scale agile in large organizations. According to the Pioneered by Dean Leffingwell, it takes the most structured approach to scale agile than scrum of scrums. It describes three levels in the organization: portfolio, program, and team. Such structure typically appeals to larger organizations, because Scaled Agile Framework (SAFe) employs a tiered approach for the delivery of work.

The large area of SAFe is related to work, called themes, map to business epics, and architectural stories.

Managing an agile portfolio

Agile can perform a large-scale portfolio with many teams and a lot of developers. One of the examples is Netflix that uses the phrase "highly aligned, loosely coupled" to describe the well-maintained agile development across a large organization.

Expanding agile practice across the organization.

The successful company that runs agile at scale level has three common factors:

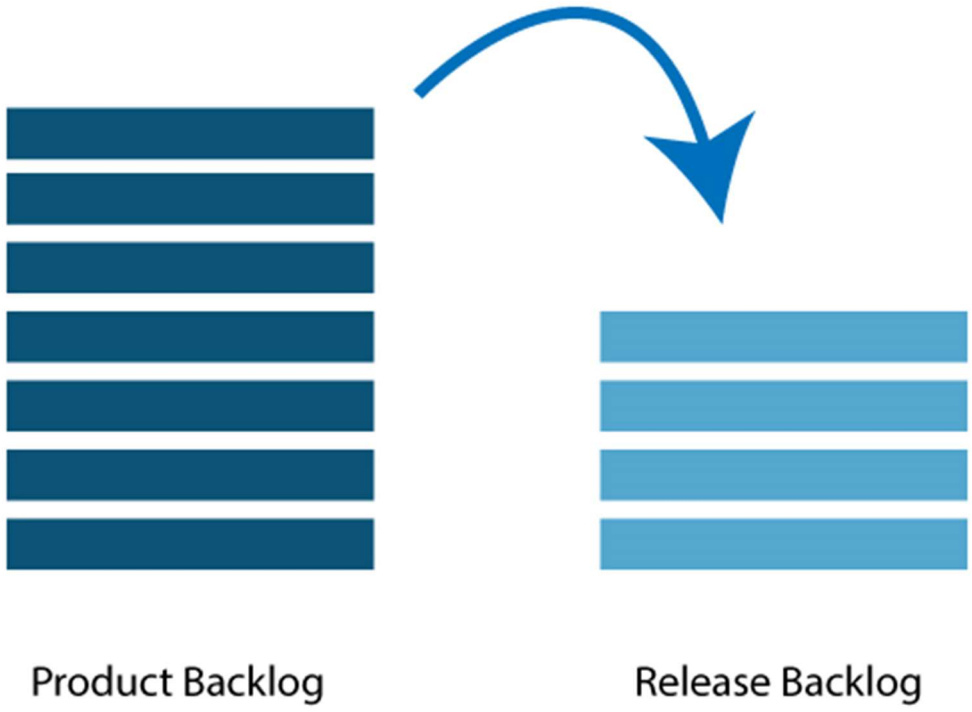
First, the entire program is iterative. The traditional is portfolio management which is focused on the top-down approach and take long periods. However, the administration takes the concept of build-measure-learn cycles for individual agile teams and applies it on a larger scale. The agile team uses modern design and share findings on a regular accent. It leads the tremendous flexibility.

Second, the organization share knowledge and break barriers between organizational silos. It also communicates across the portfolio. At the team level, similar agile ceremonies, context needs to share regularly throughout the organization so that goals, stumbling blocks, and progress are transparent for everyone.

Third, the agile company makes the frequent release product (early and often) across the portfolio, even if a release involves the work of multiple programs.

Agile Release Planning

The primary purpose of release planning is to make a plan to deliver an increment to the product. It is done in the interval of every 2 to 3 months.



Who is involved in releasing the plan?

Following person are involved in product releasing plan- Scrum Master, Product Owner, Agile Development Team, Stakeholders.

- **Scrum Master:** The Scrum Master is a team leader and facility provider who helps the team member to follow agile practices so that they can meet their commitments and customers' requirements.
- **Product Owner:** The Product Owner is one who runs the product from a business perspective. He defines the requirements and prioritizes their values.
- **Agile Development Team:** Agile development team provides the judgment on the technical feasibilities or any dependencies.
- **Stakeholders:** Stakeholders are the customers, subject matter, program manager act as advisers in decisions which are made around the release planning.

Prerequisites of Planning:

The prerequisites of release planning are as follows:

- A Product Owner manages the ranked product backlog. While releasing the product, the Product owner feels to include five to ten features at the period of product release.
- High- level vision
- Market and Business objective
- Team's input according to capabilities, known velocity, or about any technical challenge.
- Acknowledged about the new product backlog items are needed

Material Required:

The list of materials that is required for the releasing planning is as follows:

- Flip charts, markers, whiteboards/li>
- Posted agenda, purpose/li>
- Projector for sharing the data/tools of computers required during a planning meeting/li>
- Planning data

Planning Data:

The list of data needed during release planning are as follow:

- Previous iteration data or release planning requests
- Actions plans of previous release/iteration
- Features or defects to be considered
- Organizational and personal calendars
- Velocity from previous releases/ estimates

Output:

Following are the output of a release planning:

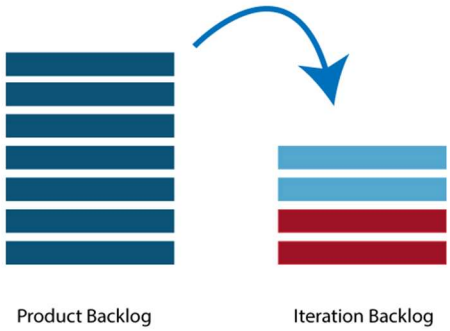
- Release plan
- Commitment
- Issues, dependencies, concerns, and assumptions which are to be monitored
- It suggests improving future release planning

Planning Agenda

- **Opening ceremony:** Welcome message, review purpose, organizing tools, and introduction to business sponsors.
- **Product Vision and Roadmap:** It shows a broad picture of the product.
- **Review previous releases:** Product planning agenda discussed on any item which can impact the plan.
- **Product release name/theme:** It inspects the current status of roadmap themes and makes the necessary adjustment if any.
- **Issues and concerns:** In the agenda, we check any concern or issue, and then record them.
- **Review and Update the Definition of Done:** Review the product build or definition of done and make appropriate changes based on technology.
- **Retrospect:** Require feedback from participants to make the meeting successful.
- **Close:** Celebrate success.

Agile Iteration Planning

The primary purpose of iteration planning is for the team. The team should be a complete set of the top-ranked product backlog items. The completion of top ranked product backlog is a commitment in the time needed on the length of iteration and team velocity.



Who involved in the iteration planning?

Scrum Master: The Scrum Master is a team leader and facility provider. He helps the team member to follow agile practices so that they can meet their commitments and customers' requirements.

Product Owner: The Product Owner deals with a complete view of the product backlog and their acceptance criteria.

Agile Development Team: Agile delivery defines their tasks and sets the effort. The effort is to estimate the requirements to fulfill the commitment.

Prerequisites of Planning

- The items in the product backlog are sized and have a relative story point assigned.
- The product owner gave the ranking to the portfolio items.
- Acceptance criteria of each portfolio item is clearly stated.

Planning Process

Iteration planning involved the following steps:

- Determines how many requirements (stories) are fit in an iteration.
- Break this requirement into tasks. Assign each task to their owners.
- Each task is set to some estimated time.
- These estimates help the team members to check how many hours for each member will be required to iterate.
- Team members are assigned tasks by seeing their velocity or capacity. Due to this, the team member is not overburdened.

Velocity Calculation

The agile team calculates the velocity based on the previous iterations. A velocity is an average number of units that required finishing user stories in the iteration. Assume that, a team took 10,12,8 story points in each iteration for the previous three iterations, this shows that the team can take 10 as velocity for the next iteration.

Planned velocity tells the team how many user requirements can be completed in the current iteration. If the team instantly finishes the work assigned, then more user requirements can be pulled in. Otherwise, the requirement can be moved out too to the next iteration.

Task capacity

Three factors determine the capacity of the team:

- Total number of ideal working hours in a day
- A person gives total days in each iteration
- Percentage of time a member is entirely available for the team.

Considered a team has 6 members, committed to work full time of 8 hours a day on a project. And no member is on leave during iteration, and then the task capacity for two-week iteration will be- 6 x 8 x 10 = 480 hours

Iteration Planning Steps

- Product Owner describes the highest ranked item of the product backlog.
- Team member describes the tasks required to complete the item.
- Team members own the tasks.
- The team member estimates their own time to finish each task.
- The above steps are repeated for all the items in the iteration.
- If any member is overloaded with work, then his/her tasks distributed among other team members.

Agile Product Backlog

The agile product backlog in Scrum is a list of prioritized features. It contains a short description of all the functionalities desired in the product. In usual scenario, items should be broken down into user stories. Commonly, a Scrum team and its product owner write everything that they can think for agile backlog prioritization.

Why Product Backlog is Important?

- The backlog is prepared to provide an estimate of each feature.
- It helps in the planning of the product's roadmap.
- It helps in the re-ranking the features of the product by adding more value to it.
- It assists in determining the priority of the product first. The team member works first on the higher prioritizes product.

Characteristics of Product Backlog

Each product should have its own product backlog. It can be a set of large to very large features.

Multiple team members can work on a single product backlog.

Ranking of product is based on the technical value, business value, risk management, or strategic fitness.

Highest priorities items are decomposed into smaller stories during release planning. This is because they can be completed in future iterations.

The Product Backlog comprises the following different types of items:

- Features
- Bugs
- Technical work
- Knowledge acquisition

Agile Tools

In agile development, leading as project management is not the easiest job. Jumping between your daily scrums to your next sprint, it causes hard to focus on the work. The agile development tools fulfill your needs, and does it for you.

There are several agile tools available in the market. Some of them are listed below:

JIRA Agile

Jira is a tool developed by Australian Company **Atlassian**. It is used for **issue tracking, bug tracking, and project management**. The bugs and issues are related to your software and Mobile apps. The Jira dashboard consists of many useful functions and features. This function and features make secure handling of issues.

Agile Software Features:

- Issue tracking
- Bug tracking
- Boards
- Epics
- Custom fields

ClickUp

ClickUp is one of the ultimate agile management tools. It is used for anyone who uses agile methodology. It is the only project management tool whose goal is "to move quickly and easily". **ClickUp** is in the hand of some of the most famous agile team, including **Google** and **Apple!** It is a free forever plan, so, the team can get their hand of ClickUp.

Agile Software Features:

- Create epics
- Use story points
- Analyze sprint performance
- Time estimates
- Start and due dates
- Time tracking

GitHub

GitHub is one of the largest hosted Git serve where the developers can store all of their codes for a vast number of projects there. The GitHub provides such a facility of record edits across an entire team in **real time**. GitHub is also integrated with many other tools so, many people such as developer and product owner can work on the same code at the same time.

The project manager can make the GitHub work for their team. It includes lots of project management tools which help him to inspect what the development team is working on.

Agile Software Features:

- Issue tracking
- Mentions
- Labels
- Link issues and pull requests

LeanKit

LeanKit is the ultimate management tool for a **Kanban** board on the agile progress for your sprints. It uses cards to represent the work items and live statuses of team member. It works perfect for the remote employees to ensure everyone can see the Kanban board in real time. It prevents the same task to complete twice and make sure the whole team remains on the same page.

LeanKit work well for cross-functional team which is benefit for Scrum or Kanban boards.

Agile Software Features:

- Board view templates
- Track issues and bugs
- Manage project portfolios
- Lean metrics and reporting

Planbox

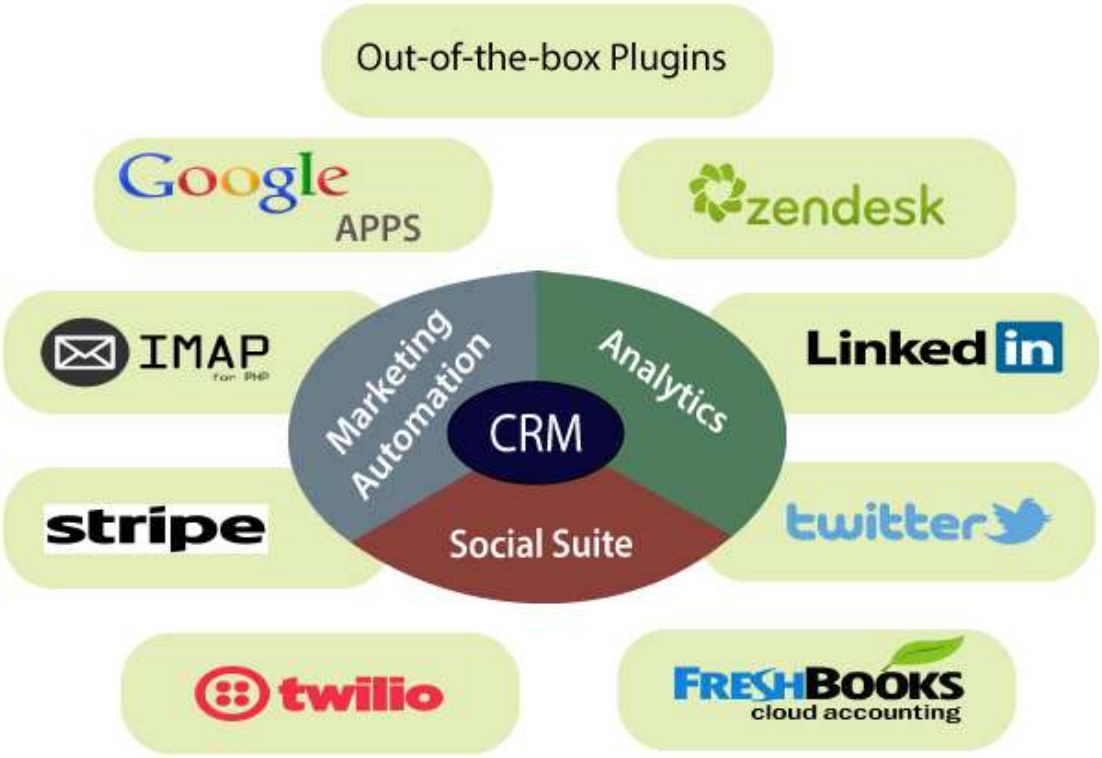
Planbox is a tool that tracks the process of burndown charts. Using this everyone knows how far you are from the Sprints completion/goal. Burndown charts are most important part of the agile cycle. Planbox also integrates the customer bug reports, and fixes, making it useful for a wide range of users.

Agile CRM

Agile CRM (Customer Relationship Management) is an All-in-One CRM with Sales, Service automation, and Marketing in a single platform. It consists of sales tracking, marketing automation, contact management, web analytics, telephony, two-way emails, and helpdesk with a simple, clean, and modern interface.

Customer Relationship Management or CRM is software that helps in managing the brand's engagement with your current and future customer.

The CRM software allows you to build, organize, and present database of your customer information. This information can be updated by you and your team when the new data is discovered. It is a central storehouse of all your customer and prospect information which facilitates your team to be organized and more productive. It also allows teamwork among teams and provides management to deeper judgment into individual performance and overall growth of the business.



Why we built Agile CRM

As an entrepreneur, everyone wants to receive positive feedback and increasing the new product's success. From the first few customers, you engage them with their name, knowing them well enough to talk to them multiple times a day. After that, they may help you to share your product to where it stands. Then, you will get more success, but more trouble also to manage the communication with all of your customers.

Ashirbad Swain

Need of Agile CRM

The primary requirement of Agile CRM is broken down into the following points:

1. **Easy-to-use marketing automation.***:* This is easy to use because not every owner of a small businesses should need to understand the technical details of automation.
2. **A manageable CRM.***:* it is easy to use and affordable.
3. **Telephony.***:* There was probably nothing more annoying than seeing someone's phone number on CRM and then dialing their number on our phone. We spent a lot of time in solving our cell phone plans too!
4. **Social suite.***:* social media playing a vital role in CRM marketing and linking customer with it. If the business owner doesn't incorporate with social media at the very beginning, then they would be left behind.
5. **Full, two-way email integration.***:* The owner wants to integrate personal emails fully into the CRM. So, they can send, receive, and view emails between the owner and their customers inside the CRM itself.

Benefit of Agile CRM

The significant benefit of CRM is that the business moves to the centralized platform to store its data. It makes easy access to information from one common source. Due to the presence of best CRM, the organization gets confident to pay attention to their customer without any additional cost.

Before the availability of CRM, the data has scattered across spreadsheets, documents, address book, notebook, and email system. The CRM, simplify this tedious process, and data are access through a centralized location.

Agile Methodology Interview Questions

1) What is an agile or agile methodology?

Agile is an iterative approach of software development methodology using short iterations of 1 to 4 weeks. Due to the agile methodology, the development process is aligned to deliver the changing business requirement.

2) What are some quality strategies of agile?

Some quality strategies of agile are:

- Iteration
- Re-factoring
- Dynamic code analysis
- Short feedback cycles
- Reviews and inspection
- Standards and guidelines
- Milestone reviews

3) What are an agile manifesto and its principle?

Agile manifesto uncovers the better way of developing software by doing it and helping others to do it. Agile has 4 manifesto and 12 principles which defines:

- Individuals and interactions, i.e., self-motivating and self-organized should be encouraged.
- Demonstrate the working software at regular intervals with comprehensive documentation.
- Customers are collaboration over contact negotiation.
- Responding to change over following a plan.

The principles of agile manifesto are-

1. **Customer Satisfaction:** Manifesto provides high priority to satisfy the costumer's requirements. Customer satisfaction is done through early and continuous delivery of valuable software.
2. **Welcome Change:** Making change during software development is common and inevitable. Every changing requirement should be welcome, even in the late development phase. Agile process is used to increase the customer's competitive advantage.
3. **Deliver the Working Software:** Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest period.

4) Is there any disadvantage of the agile model (SDLC)?

Disadvantages of Agile SDLC:

- The development team should be highly professional and client-oriented.
- New requirement may be a conflict with the existing architecture.
- With further correction and change, there may be chances that the project will cross the expected time.
- There may be difficult to estimate the final coast of the project due to constant iteration.
- A defined requirement is absent.

5) What is the burn-up and burn-down chart?

The burn-up chart depicts the amount of work done in the project, whereas the burn-down chart illustrates the amount of work remaining in the project. Thus, the burn-up and burn-down are used to describe the progress report of the project.

6) What do you understand by Daily Stand-Up?

The daily stand-up is the day-to-day meeting (mostly in the morning) in which the whole team meets around 15 minutes to find the answer for the following three questions:

- What was done yesterday?
- What is your plan for today?
- Is there any obstacle that restricts you to complete your task?

7) What do you understand about Scrum?

Scrum is a framework that helps agile teams work together to develop, deliver, and sustain the complex product in the shortest time. The product provides by scrum team in this shortest period is known as a **sprint**.

8) What are the different roles in Scrum?

There are three different roles in scrum. These are the *Scrum Master*, *Product Owner*, *Agile Development Team*.

- **Scrum Master:** The Scrum Master is a team leader and facility provider who help the team member to follow agile practices so that they can meet their commitments and customers' requirements.
- **Product Owner:** The Product Owner is one who runs the product from a business perspective. He defines the requirements and prioritizes their values.
- **Agile Development Team:** Agile development team provides the judgment on the technical feasibilities or any dependencies.

9) What are the responsibilities of the Scrum Master?

The critical responsibility of Scrum Master includes:

- Tracking and monitoring project development.
- Understanding the user requirement correctly.
- Work to obtain the project properly.
- Improving the performance of the team.
- Organized meetings and resolve issues.
- Communicate and report to the customer and development team.

10) What are different ceremonies and their importance in Scrum?

To clearly express the Scrum planning, Scrum review, Scrum Daily stand up, and scrum retrospective is the purpose of the ceremony. The importance of these ceremonies is to use sprint as per your project.

11) What do you know about Scrum ban?

Scrum-ban is a Scrum and Kanban-based model for software development. This model is used in the project that needs continuous maintenance, various programming error, or some sudden changes.

12) What do you understand by the term agile testing?

The agile testing is the software testing process which is fully based on the principle of agile software development. It is the iterative approach where the user story becomes the output of the collaboration between the product owner and the development team.

13) What are the major principles of agile testing?

Some of the essential principles of agile testing are:

- Customer satisfaction
- Face to face communication
- Sustainable development
- Continuous feedback
- Quick respond to changes
- Successive improvement
- Self-organized
- Focus on essence
- Error-free clean node
- Collective work

Ashirbad Swain

14) What are the skills of a good agile tester?

The agile tester is one who implements the principle of agile software development principles for software testing. An excellent agile tester has the following skills:

- He must be familiar with the principles and concept of agile.
- He must be excellent communication skill to communicate with the team and the clients.
- He can set the priority of a task according to customer requirements.
- He should able to understand the customer requirement properly.
- He should understand the project risk due to changing demand.

15) Name the agile frameworks.

Some of the agile frameworks are:

- Scrum
- Kanban
- Feature Driven Development
- Test Driven Development

16) Is it ever suggested to use waterfall over Scrum? If yes, explain when.

Yes, sometimes we use waterfall module over scrum. This is because when the client requirement is simple, small, well-defined, fully understood, predictable, and the subject does not change until the project complete.

17) Name some methodologies and development where you have used the agile model.

While answering this type of question, keep in mind to mention those methodologies from which you are familiar whit. Some of the methodologies where agile is used are:

- Crystal methodologies
- Lean software development
- Dynamic development
- Feature-driven development

18) What was the length of sprints/iterations in your project?

It is a common question for experienced people. The idea behind is to judge in which kind of environment you have worked? There will be follow up of the question that the length fixed in the beginning and never changed? Did you try with less than this length or more than that?

19) What is the difference between the agile & traditional way of working?

The traditional way of development is that which follow the sequential where design -> development -> testing etc. is performed whereas, in agile development, all of this is done in every iteration/sprint.

20) Why does Scrum encourage the use of automated testing for projects?

Due to faster possible delivery of the project, the Scrum development encourages to use automated (automated performance or automated regression) testing. While answering this question, you should explain some tools that you have used for automated testing.

SDLC Concept

The process by which the software is conceptualized, developed and maintained is known as software development lifecycle or the SDLC.

There are seven key points to understanding software development lifecycle, phase 1 is planning phase, phase two requirement analysis phase, phase 3 design, phase 4 is implementation and coding phase, phase 5 is testing phase, phase 6 is deployment and phase 7 maintenance.

So, it all starts with this guy, the customer. he's guy that has business idea for our invoice application and the money to get it started. He's going to reach out to multiple different tech companies until he finds one that he likes.

Finally, he is going to meet our company's product owner or project manager. They are going to discuss terms of their agreement, sign a deal and accept the project. This will move us into the next phase i.e., planning the requirements.

Together the customer and the product owner will outline the requirements of the application.

Let's imagine that the requirements that they have both agreed upon are as follows:

Defining the Application

1. User Registration
2. Login
3. Logout
4. Dashboard loading page and so on and so forth

Now that we have outlined requirements. Let's take them and move on to the requirement analysis phase.

Here are our team operations developers, product owners and testers. We will all meet up in an office for a few hours and defined each outlined requirement and give more planning details.

Let's start with the requirements:

Defining the Requirements

For the User Registration will need,

1. A username input field
2. A password fields
3. A checkbox to accept the terms and conditions
4. A submit button
5. The ability to save the user into our database

Then we'll move to requirement number two: Login, we need a login page to allow returning users to log back into our system, for this we need:

1. Another username input field
2. A password fields
3. along with another submit button
4. from this we'll need to read the users information out of the database
5. and log them into our system

Then requirement number three will require a logout button,

1. logout button usually located around the same area where the user was logging in from when they press the logout button
2. a clear session for clear their session out of the browser to prevent other people from returning and logging back into their account
3. prevent account theft

finally, requirement number 4 the Dashboard,

1. this will essentially be the main home page of our application
2. after a new user is registered, they should be redirected to the dash board (new user should redirect here)
3. also, when the user logs in with the returning account, they should be redirected to the dashboard for a first SDLC. (Existing users redirect here after login)

Once all of the requirements have been analyzed by the team and defined, the product owner will take all the defined requirements and create tickets in a project management system.

From here we'll move into the design phase of the SDLC. The design phase takes all the requirements and starts to plan the product. The design phase may include the business rules, the user interface layouts, color schemes, what programming languages to use, frameworks, system server design, database relationships, architect of the application, mobile aspects, supported browsers and much more.

Next, we will discuss the implementation and coding of the application.

The implementation and coding phase is where everything starts to become fun, the operation team will set up the physical hardware for the servers, the developers will start writing the code, the designers would continue planning the user interface and the tester will analyze the requirements and start building test cases for the test plans. Even in this stage testers are incredibly valuable; they start to imagine the usability of the application and see how everything flows together.

Sometimes while writing test cases, they can discover things don't make sense and help free design fundamental flaws in the early stages of the application.

Next up we'll be discussing one of the most important phases which is the testing phase.

So, why is the testing phase being so important.

Imagine the developers has finished coding some of our new features in our requirements we found that when the user logs out then it's not actually clearing the session in the browser and that another person can walk up to their computer click the login button or click the refresh on the page and they gain access to their account, get access to all of their money and start making payments on various things it would be huge and expensive terrible bug that would get released if we didn't have testers to test these types of things beforehand. So, what does the tester actually do during the testing phase.

Now that we have the servers all set up and the database is set up, developers have finished coding. They've given us an application, an actual website that we can log into now. Now we can start testing and executing that our test cases from the test plans. That we have created.

Validate that all of the requirements have been met, make sure all the functionalities are working as expected, find as many bugs as we possibly can which could be color scheme is incorrect or there's a user interface bug, somewhere the critical issue people not being able to logout, may be the users can't even register, there's so many mistakes that the developer can make and do make free code.

So, as we are testing, we start to find bugs and what we'll do is, we'll report them into a bug tracking system which is then assigned to a developer. They'll go in and they'll fix the bug, fix the issue in the signup back to us, this is called a bug lifecycle.

Up next, we will talk about the deployment phase,

In the deployment phase, the operation team will end up nearing the staging or development environment systems that we've testing in and get them ready for production meaning that they'll install new hardware or brand-new servers have everything scalable for production, this includes setting up the links, setting the databases for real users seeking up with the development teams and release managers once they've completed all of these tasks our application will go live to real users.

This brings us to the final phase of the SDLC i.e., Maintenance.

So, imagine that we've released ad our application became so successful. We're just getting millions of users logging in and registering and using this application, so we need to maintain the servers in the environment.

They need to monitor the load, the stress everything coming on the server's mine, so many users logging in and using it. It doesn't bring down the system maybe we need to make larger servers, larger databases maybe we need to get fasters computers, there's a lot stuff that goes on during the maintenance phase.

There will be bugs found in production. It's called production support. Frequently users will email with their issues and you can stop and investigate what they're complaining about or what their issue is. Figure it out write up a bug get it resolved and do another deployment to production with issues fixed.