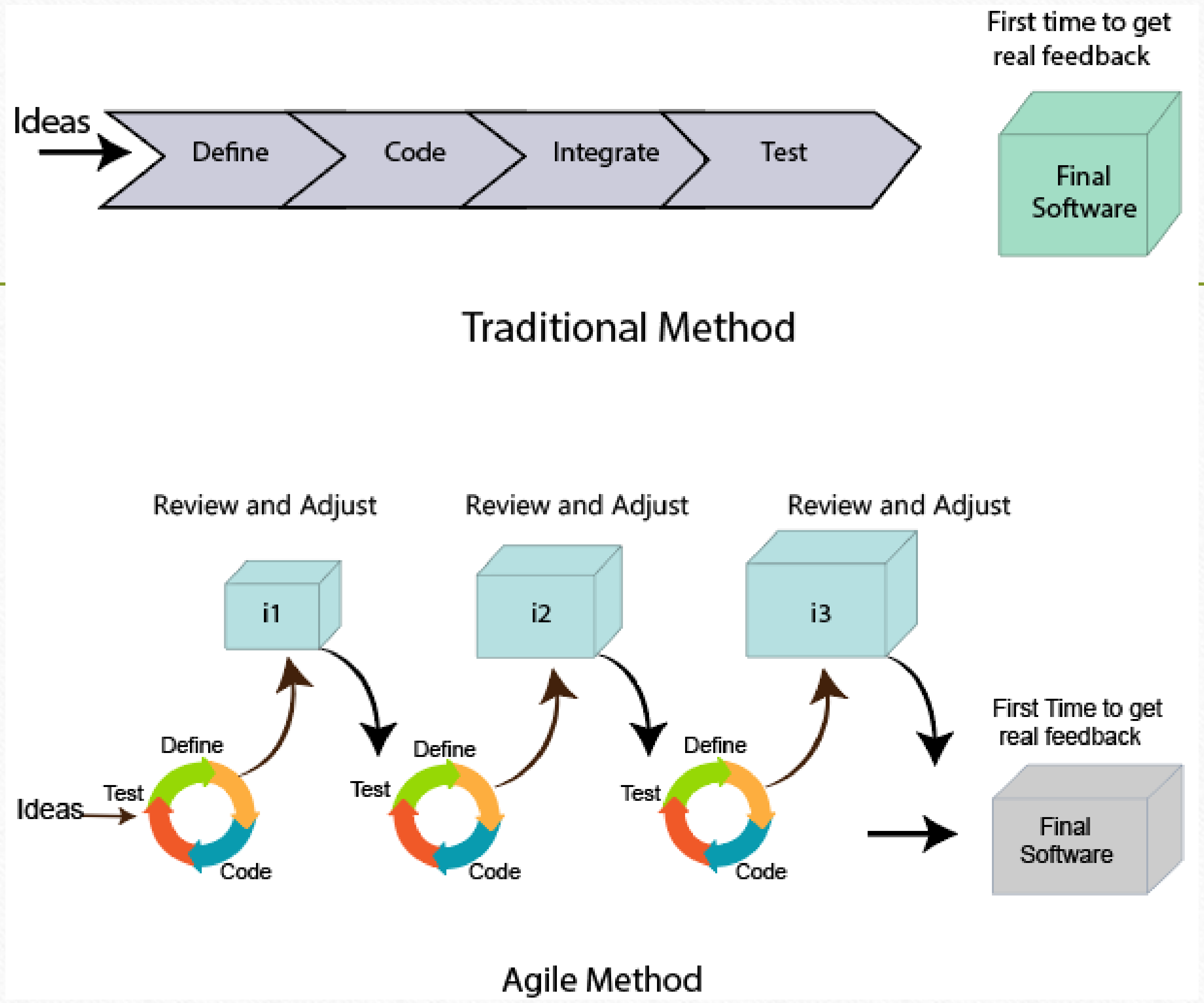
Chapter 3

Agile Project Management Framework

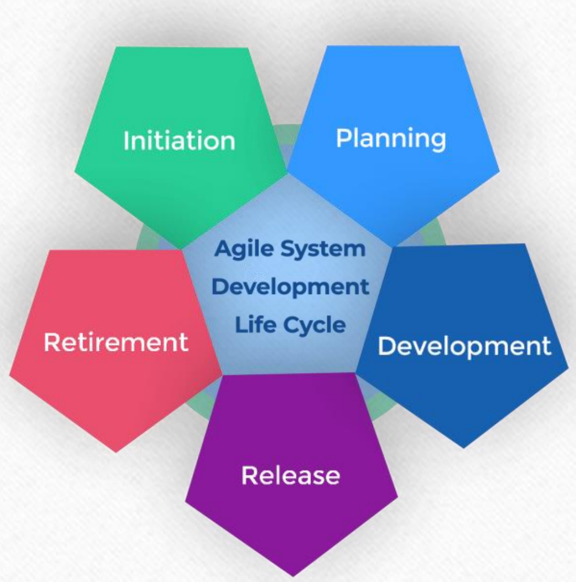
What is Agile Methodology?

- 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. The agile development process is aligned to deliver the changing business requirement. It distributes the software with faster and fewer changes.
- The single-phase software development takes 6 to 18 months. In single-phase development, all the requirement gathering and risks management factors are predicted initially.

- The agile software development process frequently takes the feedback of workable product. The workable product is delivered within 1 to 4 weeks of iteration.


First time to get real feedback
Ideas
Define
Code
Integrate
Test
Final Software
Traditional Method
Review and Adjust
Review and Adjust
Review and Adjust
i1
i2
i3
Define
Test
Code
Ideas
First Time to get real feedback
Final Software
Agile Method

Agile Life Cycle



1. Project Initiation:- The first stage in the life cycle of agile software development. Often referred to as the inception or envision phase, this initial stage is about discussing the project vision. During this step, you should identify team members and determine the time and work resources are required to complete the project.
2. Planning:-This phase is when the Agile lifecycle really takes shape for the team. Release planning is where the team gets together with their sponsor or product owner and identifies exactly what they are looking for. They discuss how this will be made possible by building the backlog at the story level. The end-user might describe the feature or product, type of user, what they want from the product and why

You should be estimating the risks and developing milestones with an initial release plan. Planning is only complete when your backlog is complete and you have prioritized the items based on business value and dependency

3. Development:- Once the requirements have been defined based on feedback from the product owners and stakeholders, the actual work begins. Agile product development delivers high quality working products in incremental phases, sprints, or iterations. Developers start building the first iteration of the product with the aim of having a working, usable product at the end of the sprint.

Teams can deliver in these sprints by:
Ensuring Collaboration with their team and stakeholders.
Maintaining quality by following coding conventions and style guidelines.
Adhering to priorities set by stakeholders who are given total control of the scope, budget, and schedule of the project.
Delivering working products however limited, at the end of every cycle.
Testing all the time!

4. Production:-Your product has now been deployed and is being used by final end-users. It is important to closely monitor these early stages for bugs or defects missed in testing. A handover with relevant training should take place between the production and support teams.

5. Retirement:- The final stage of the Agile lifecycle. The product is now at the 'end of life' stage and will be pulled from production and decommissioned It could also be retired because the product is not cost-effective within the current business model and therefore phased out.

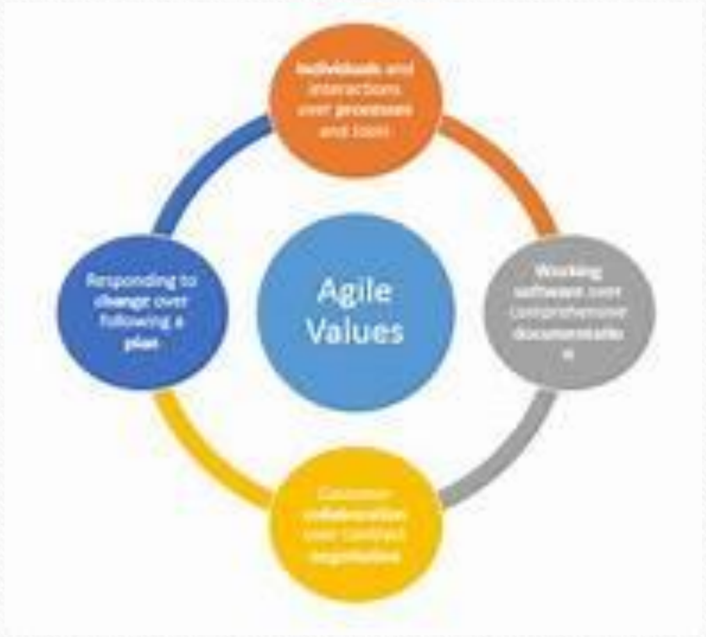
Agile History: The Starting Point

What is the Agile Manifesto?

- Agile Manifesto is the primary source to turn to when we are lost, or we want to find out how to fix a particular problem within the framework.
- The purpose of the Agile Manifesto is simple – to give every person interested in this framework a clear vision of what they should focus on and consider. They were written by software development experts that were behind the whole Agile movement. This way, we can be sure that they realistically constructed their rules to meet the requirements of the process.

What are the Agile Manifesto values?

- Agile Manifesto has its own structure – the central part contains 4 main values that are the key to using Agile methodology with success. The Manifesto's additional part includes 12 principles that are the more detailed guidelines for Agile software development.


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

The primary 4 values that are defined in Agile Manifesto are:

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

Individual and interaction over using process and tools:

1. The success of the team is determined by the ability to communicate effectively and efficiently. Other things such as the tools and the processes used are of less importance.

- The contacts among team members enable them to work together and resolve any possible issues.
- Individually, communication is flexible and takes place as needed. Process-related communication is scheduled and necessitates particular content.

2. Working software over comprehensive documentation:

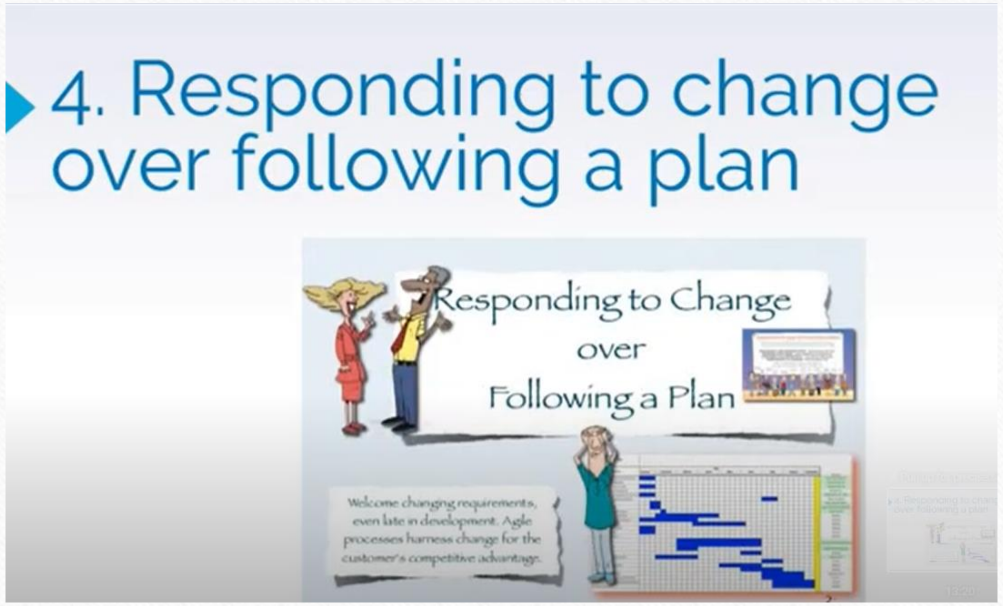
- The conventional methods for software development required very extensive documentation before any actual code is written. Focusing more on documentation and not shipping code will lead to never getting feedback in the real world.
- The agile method suggests prioritizing shipping code regularly while gathering feedback from the customer and improving upon it in the future version of the product.
- It’s vital to remember that documentation in and of itself is not a negative thing (overdoing should not happen here).

3. Customer collaboration over contract negotiations:

- The traditional methods focused on the product allowed contracts to control what was ultimately delivered, which created a lot of room for unrealistic expectations. Before any work begins, clients negotiate the requirements for the product using development methods like a waterfall, frequently in great detail. This meant that the customer was involved in the development process only at the beginning and end, not in the middle.
- The agile way of doing this is to incorporate a continuous client feedback loop into development cycles.
- Customer collaboration occurs regularly during the development process and starts early in the agile concept.
- By regularly communicating with customers and incorporating their comments into your development process, you lower risk and do away with guessing.

4. Responding to change over following a plan:

- There is nothing constant except change and static roadmaps lead to a lot of difficulty in adopting new changes and it will soon become outdated.
- Therefore the agile mindset is to go with a flexible roadmap that takes it into account, a software team should be able to pivot and change course as necessary.
- Agile teams can adapt to the changes that a dynamic plan may undergo from quarter to quarter or even from month to month.


4. Responding to change over following a plan
Responding to Change over Following a Plan
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

The 12 Agile Principles

- **1. Satisfy Customers Through Early & Continuous Delivery :-** ***"our highest priority is to satisfy the customer through early and continuous delivery of valuable software"***
- **2 Welcome Changing Requirements Even Late in the Project :-** ***"welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage".***
- **3 Deliver Value Frequently:-** ***"deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale".***

- **4 Break the Silos of Your Project :-** Agile relies on cross-functional teams to make communication easier between the different stakeholders in the project. As the original text states, ***"business people and developers must work together daily throughout the project".***

- **5 Build Projects Around Motivated Individuals :-** "***build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done".***

- **6 The Most Effective Way of Communication is Face-to-face** :- ***"The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.“***
- **7 Working Software is the Primary Measure of Progress:-** ***The 7th of the Agile core principles is pretty straight forward. It doesn't matter how many working hours you've invested in your project, how many bugs you managed to fix, or how many lines of code your team has written. If the result of your work is not the way your customer expects it to be, you are in trouble.***
- **8 Maintain a Sustainable Working Pace :-** ***"Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."***

9. Continuos Excellence Enhances Agility :- *"continuous attention to technical excellence and good design enhances agility".*

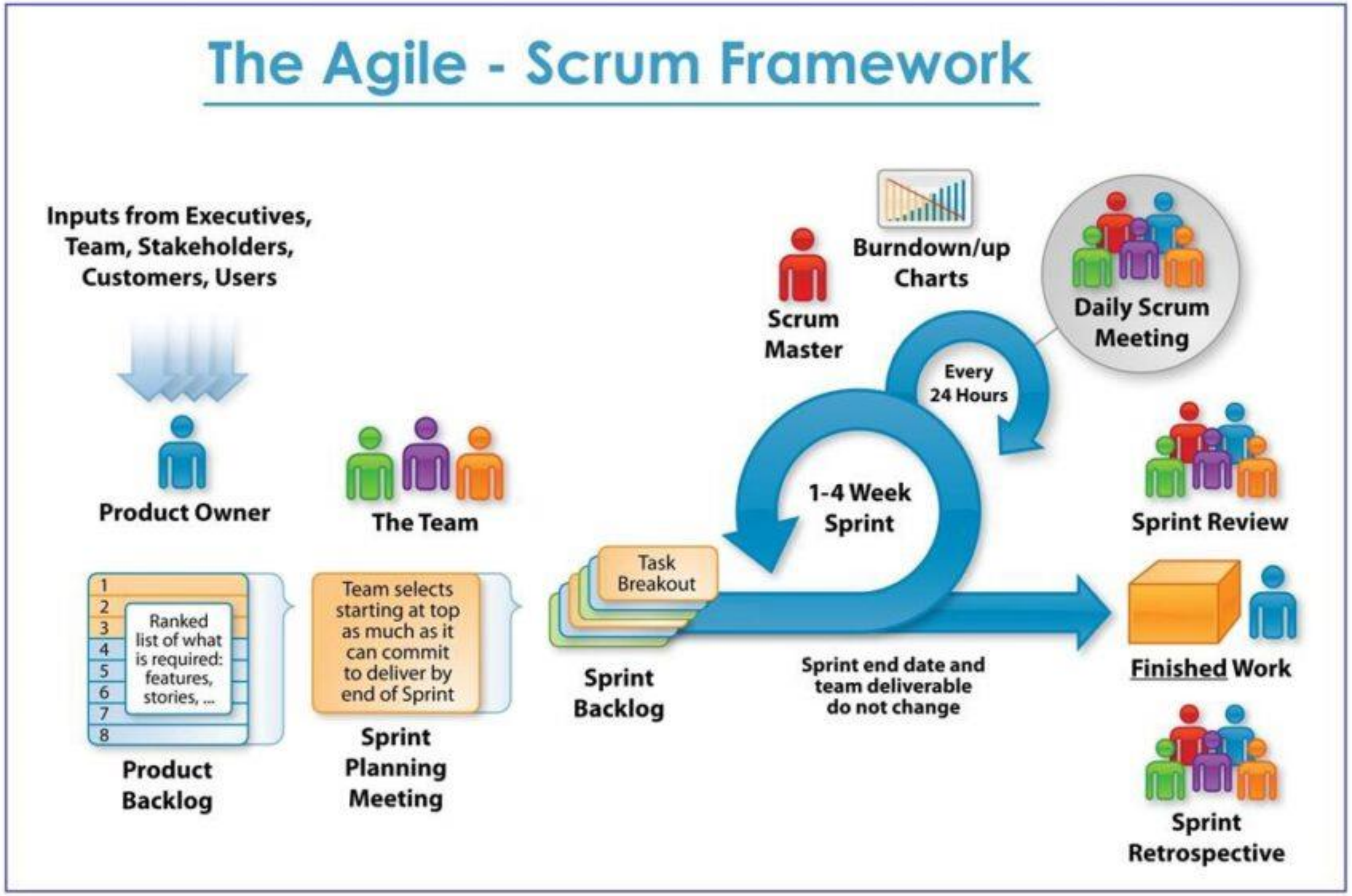
10. Simplicity is Essential :- *"Simplicity–the art of maximizing the amount of work not done–is essential".*

11 Self-organizing Teams Generate Most Value :- *"the best architectures, requirements, and designs emerge from self-organizing teams".*

12 Regularly Reflect and Adjust Your Way of Work to Boost Effectiveness :- *"At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly".*

Agile Methodology

- 1. Extreme Programming
- 2. Kanban
- 3. Lean
- 4. Scrum
- 5. Crystal


The Agile - Scrum Framework
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

How Does Agile Work?

- Here is the working of Agile -
- Define the project: The team, along with the customer, defines the project's goals, objectives, and requirements.
- Create a backlog: A backlog is a prioritized list of tasks that need to be completed. The customer, product owner, and the team work together to create the backlog.

Plan the sprint: The team plans the sprint by selecting the highest-priority tasks from the backlog and determining how much work can be completed in the upcoming sprint.

Execute the sprint: The team works on completing the tasks planned for the sprint, with daily meetings to check progress and address any issues.
Review and demo: At the end of the sprint, the team demonstrates the completed work to the customer and gets feedback.

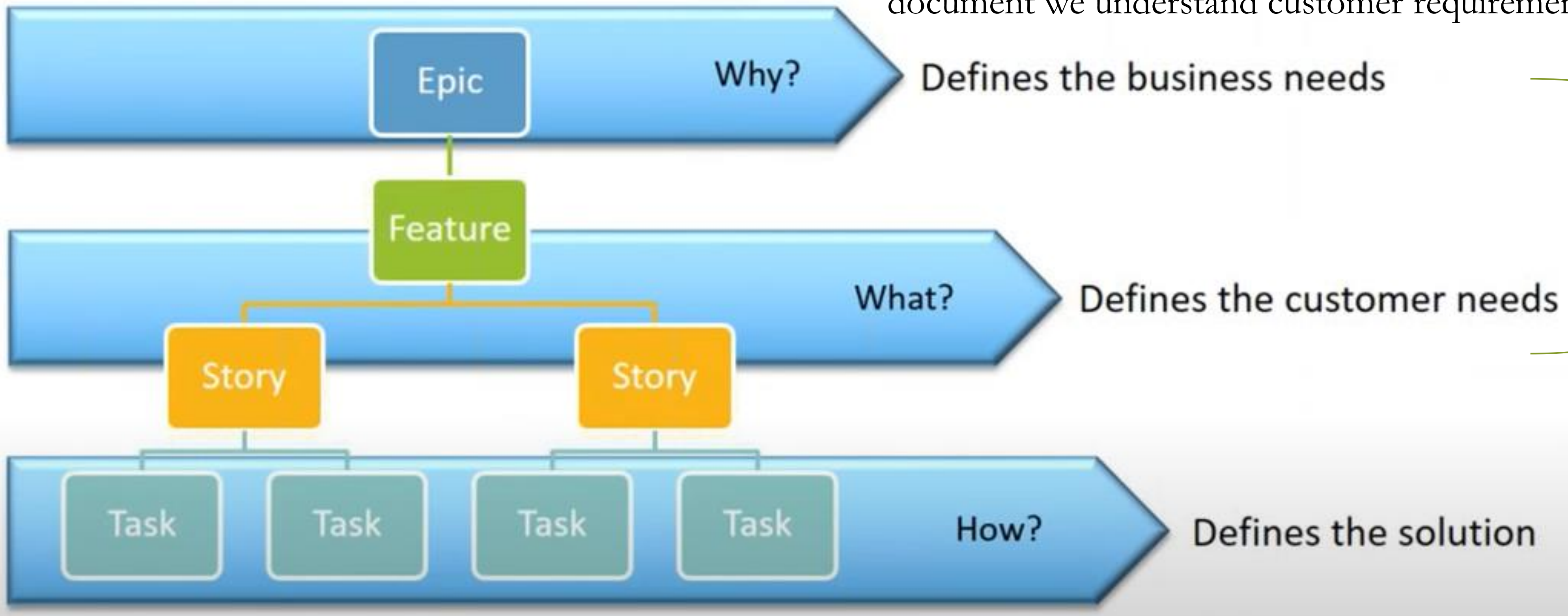
Retrospect: The team retrospect's on the sprint, discussing what went well, what didn't, and what can be improved for the next sprint.
Repeat: The process is repeated for each sprint until the project is completed. The product is incrementally developed and delivered to the customer in small chunks.

Continuously improve: Agile methodologies focus on continuous improvement. The team reflects on its progress and makes adjustments as necessary to improve processes, tools, and communication for the next sprint.

Agile Terminologies

Epic, Story & Task

In traditional model we have functional document (requirement specify by the customer so using functional document we understand customer requirement)



Product Owner

Task is Particular action perform on the story

In agile very less documentation .90% document less. So instead of FRS document we have EPIC, Story & Task

EPIC:- High level Requirement(Business Requirement define customer) Epics An epic is nothing more than a big user story. Often, they are described as ‘over-sized’ user stories, because they are too large to complete in one sprint or iteration. They are a higher-level story that is really a group of closely related user stories. So, epics are usually split into user stories that represent smaller product features. There is no definitive threshold to distinguish a big story from a small epic.

Example:- As user I need used online banking application.

STORY:- Smaller requirement(User stories A user story describes something a user wants. It is a simple description of a product requirement, in terms of what the product must be able to do…. And for whom.)

1. As user I can login into application.
2. As user I can check my balance in the application

Activities :- as user we can login, Check balance, Transfer money all are story

Story 1:- As user I can login into application.(Developer Task)

- Review the Story
- Estimate the Story
- Design
- Code
- Unit Testing
- Integration

Story 1:- As user I can login into application (Quality Assurance Task)

- Review of the story
- Test cases
- Test Scenario
- Test Data
- Review
- Test Environment
- Execute the Test cases
- Report Bug

So all the development & Testing activity perform simultaneously once the both activities completed demo given to the Product owner & after that giving to the customer or stockholder. **Collection of user Stories form the Product Backlog**

They are usually documented on cards that typically contain:

- Title
- Unique Id
- Estimate- Work Required(usually in Story Points)
- Description- what the feature needs to achieve
- Task ID- Maybe a list of tasks (like WBS elements) that are needed to create the product or feature

Definition of Ready DOR Vs Definition of Done DOD

DOR

- User Story defined
- User Story Acceptance Criteria defined
- User Story dependencies identified
- User Story sized by Delivery Team
- Scrum Team accepts UE artefacts Performance criteria identified, where appropriate Person who will accept the User Story is identified

DOD

- Code is complete and according to development team standards.
- Code refactored. Meet acceptance criteria.
- Code checked-in to the repository.
- Unit test written and green.
- Test coverage: %.Pair programming.
- Peer review.
- Code merge and tagged.
- Deployed to the development environment.

Agile Roles

In Agile process we have 3 different role

Product Owner:- Product Owner is responsible for getting requirement from customer.

he is not customer also called as Stakeholder.

1) Define the features of the product a s per the customer requirement he define the features :- stories, Epic
2) Decide release date & content
3) Prioritize feature according to mkt

   e.g of customer give to requirement then give priority.
4) Adjust features & priority every iteration

   example:- we deliver product version wise.
5) accept reject work Result –

   e.g- after development & testing work together & complete product forward to product owner so PO check the product work as per Customer Requirement If he not satisfy Reject work.

User Stories

User stories are a key component of agile software development. They are short, simple descriptions of a feature or functionality from the perspective of a user. User stories are used to capture requirements in an agile project and help the development team understand the needs and expectations of the users.

In Agile software development and product management User Story refers to a short, informal, and simple description of software features that are required by the end-users in the software system. Its main purpose is to provide software features that will add value to the customer requirements.

Pattern of User Story:
User stories are completely from the end-user perspective which follows the Role-Feature-Benefit pattern.

```
    As a [ type of user ], I want [ an action ], so that [ some reason ]
```

Principle of User story :

A good user story should be based on INVEST principle which expresses the quality of the user story because in base a good software product is completely dependent upon a good user story. In 2003 INVEST checklist was introduced by Bill Wake in an article.

Independent –

Not dependent on other.

Negotiable –

Includes the important avoid contract.

Valuable –

Provide value to customer.

Estimable –

It should be estimated.

Small –

It should be simple and small not complex.

Testable –

It should be evaluated by pre-written acceptance criteria.

3 C’s in User Stories :

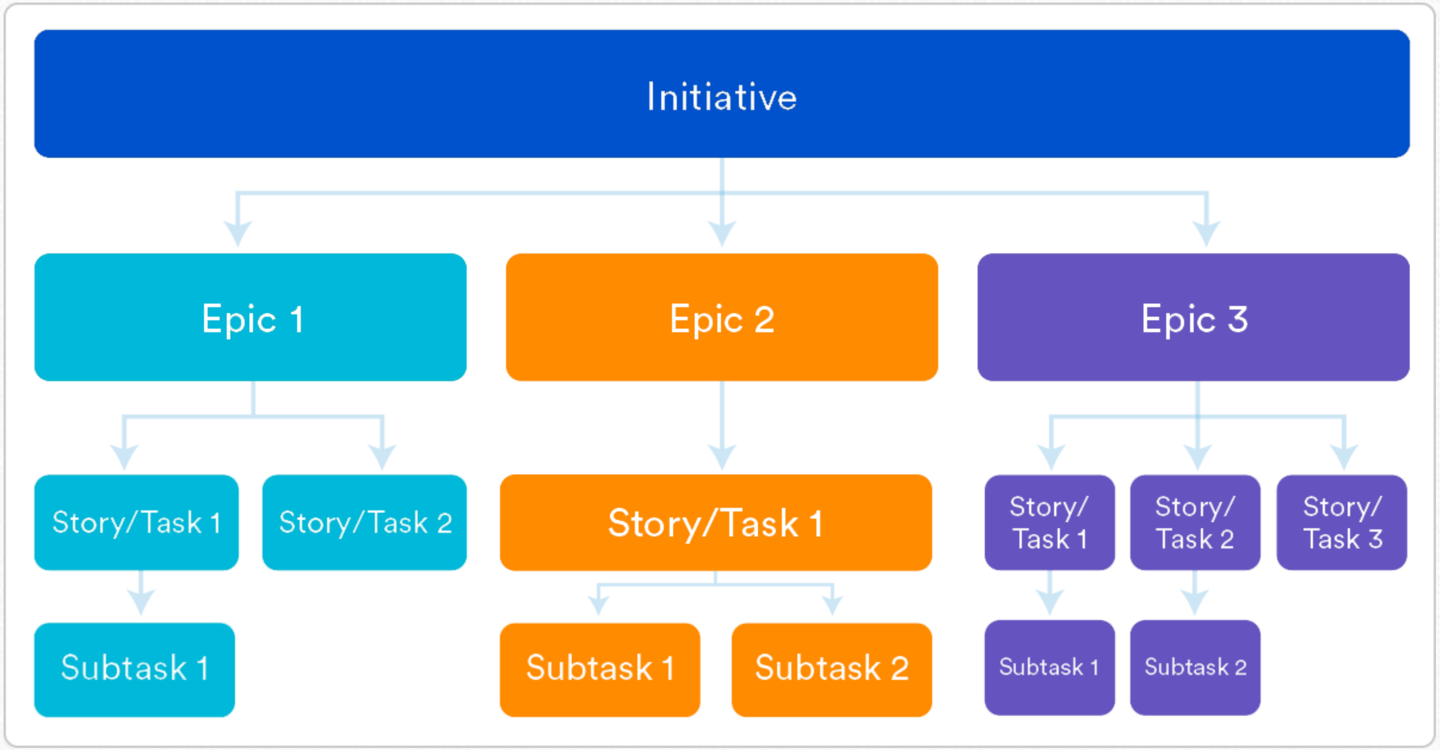
1.Card –
Write stories on cards, prioritize, estimate and schedule it accordingly.

2.Conversation –
Conduct conversations, Specify the requirements and bring clarity.

3.Confirmation –
Meet the acceptance criteria of the software.

Importance of creating User stories :

- Stories clear idea about requirements
- Makes it easy to understand the features
- Delivers higher customer satisfaction
- Fasten development process
- Creates an effective work environment
- Enables collaboration between teams
- Delivery of valuable software



User stories are also the building blocks of larger agile frameworks like epics and initiatives. Epics are large work items broken down into a set of stories, and multiple epics comprise an initiative.

What are story points?

- Story points are units of measurement used to determine how much effort is required to complete a **product backlog** item or any other piece of work. The team assigns story points based on the work's complexity, amount, and uncertainty.
- Building off the definition of an Agile story, story points are metrics used in Agile product development and management to guess how difficult it will be to implement a user story. Put another way, it’s a numeric value that helps the development team understand how challenging the story is.

Agile story points are usually represented using the Fibonacci sequence. In this sequence, each number is the sum of the two preceding numbers. It looks like this:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144…

Story point estimation
There are three key factors in the story point estimation process:
Complexity: How difficult is the user story? Does it require a lot of steps to complete?
Risk: What are the potential risks involved? These could include uncertainty or dependence on a third party.
Repetition: How monotonous are the tasks? When a team member is familiar with certain tasks, the complexity and risk factors are reduced.

All of these elements must combine for accurate story point estimation.
A product owner will be involved in the story point estimation process, but they will not estimate Agile story points themselves — this is a team effort. The Agile team members will be the ones working on the user story, so they are better equipped to estimate the effort required. Instead, the product owner will focus on sprint planning and prioritizing the backlog, which is a list of items that the development team needs to work on.

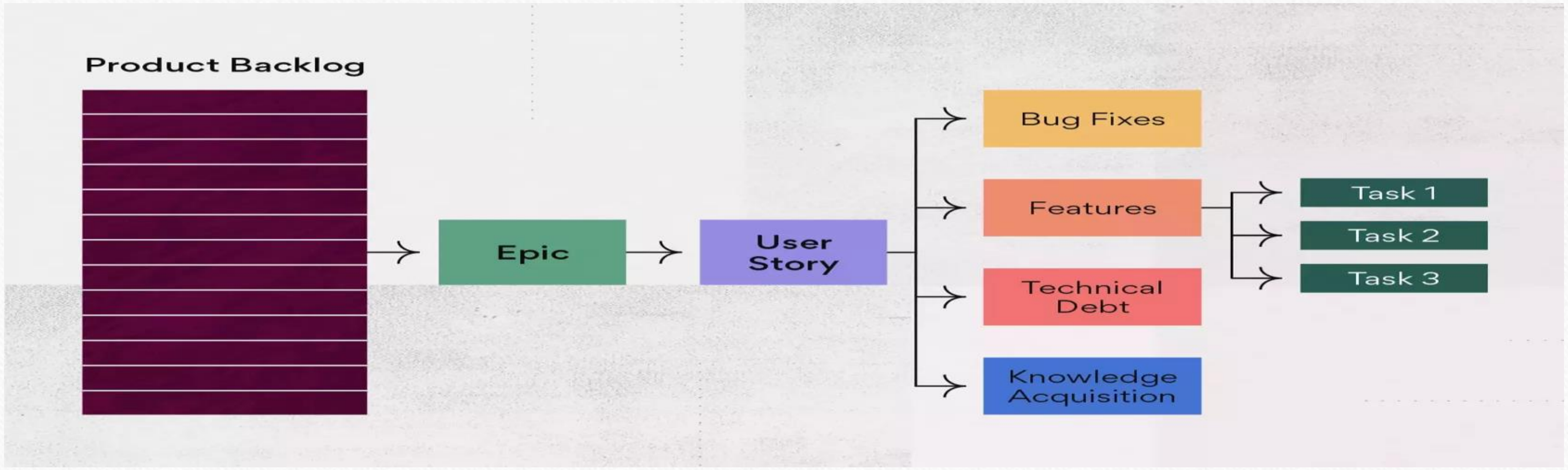
Product Backlog

- A product backlog is a prioritized list of work for the development team that is derived from the roadmap and its requirements. The most important items are shown at the top of the product backlog so the team knows what to deliver first. The development team doesn't work through the backlog at the product owner's pace and the product owner isn't pushing work to the development team.
- A team's roadmap and requirements provide the foundation for the product backlog. Roadmap initiatives break down into several epics, and each epic will have several requirements and user stories.

Who uses product backlogs-- While any developer can use a product backlog, they're most often used by Agile teams.

A product backlog tracks what the product team works on. Depending on the size of your organization, you may have one central product backlog or multiple product backlogs for different teams. The product owner will refine the product backlog periodically to make sure the most important initiatives are at the top and each initiative has all of the information needed to execute against it.

<u>What's in a product backlog?</u>



Product Backlog Refinement

To respond to changes and adapt to an agile framework, agile teams constantly update their product backlogs. That's known as backlog grooming or backlog refinement. It consists in adding, deleting and prioritizing tasks in the agile product backlog to maximize the efficiency of an agile workflow.

What is a Sprint Backlog?

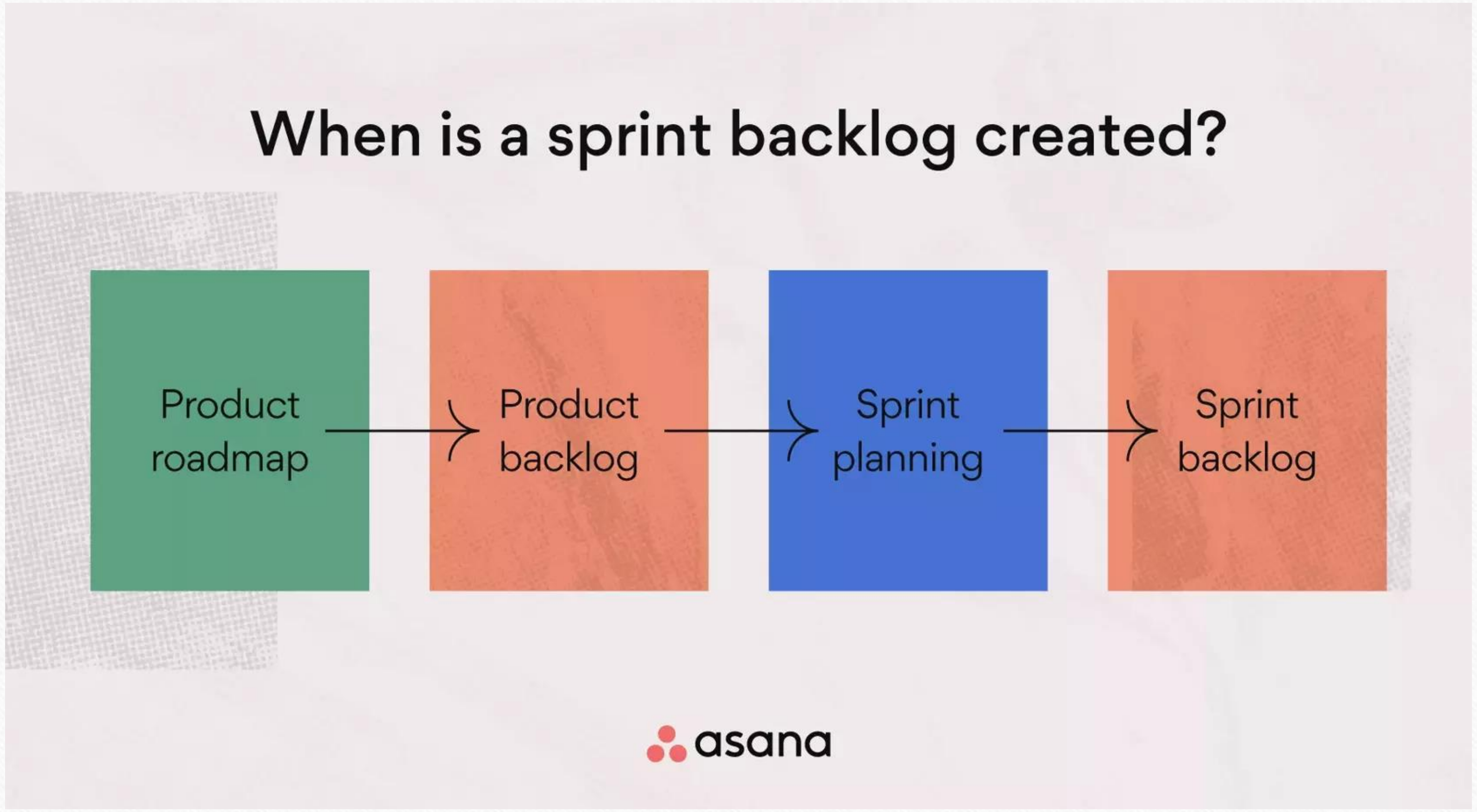
A sprint backlog is a subset of the product backlog and lists the work items to complete in one specific sprint. The purpose of the sprint backlog is to identify items from the product backlog that the team will work on during the sprint. This occurs during the sprint planning process. These items move from the product backlog into the sprint backlog and shouldn't change once the sprint begins.

A sprint backlog is the set of items that a cross-functional product team selects from its product backlog to work on during the upcoming sprint.

The purpose of a sprint backlog-- The purpose of a sprint backlog is to define work items to tackle within the sprint.

When is a sprint backlog created?

Create a sprint backlog during the planning phase of a new project sprint.



Sprint Velocity in agile

- **Velocity** in **agile** development measures the quantity of work a team can accomplish in a **sprint**
- What is sprint velocity?-- Velocity is the number of story points completed by a team in one Sprint. Some teams use different measurements, like hours or stories completed, to calculate their velocity. Whatever data you use, the concept remains the same: Velocity indicates how much work the team has finished in a Sprint.

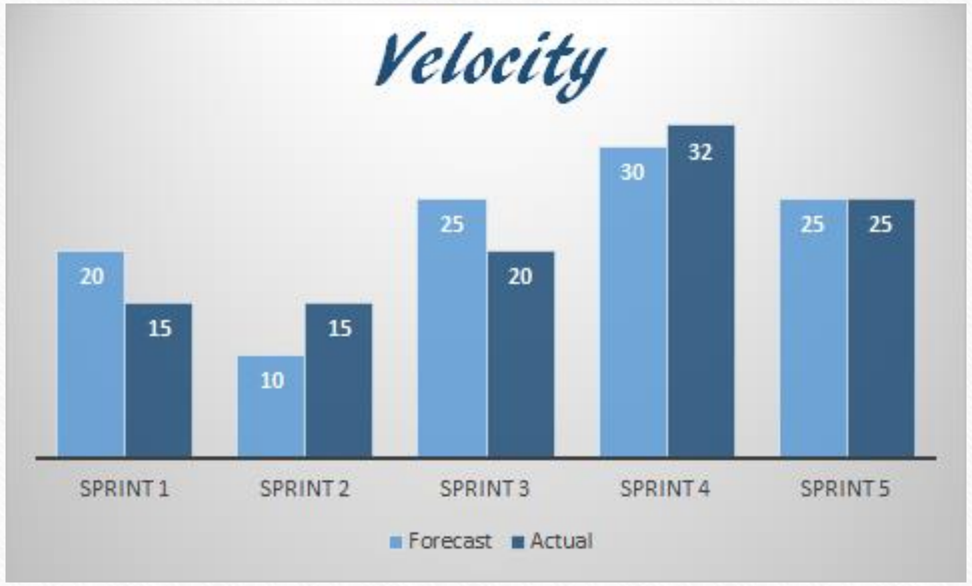
How to calculate sprint velocity

- You can calculate sprint velocity with a simple math equation: divide the number of backlog items (or story points, if that’s what your team uses) by the total length of your sprint in days.
- For example, if your team has 60 backlog items, and you average sprint duration lasts 2 weeks, the equation would look like this:
- 60 backlog items/10 days = Sprint velocity of 6

Why measure sprint velocity?

- **Makes sprint planning easy.** -- For product owners and Scrum masters, knowing your team's sprint velocity can help make sprint planning easier. If you know your team's average sprint velocity, it's easier to choose the right user stories from the product backlog to move into this iteration without overloading your development team.
- **Manage stakeholder expectations. --** If your stakeholders are asking for a timeline on a specific user story, or if they're trying to add anything before the end of the sprint, you as a product owner understand how that change may affect your team's output based on their sprint velocity.

Signals potential trouble. When you regularly track sprint velocity, you'll be able to measure the average velocity more consistently. If you see a sudden dip in velocity, then you know there's a potential issue, such as a roadblock like an unfinished dependency, that needs to be resolved before moving on to the next sprint.



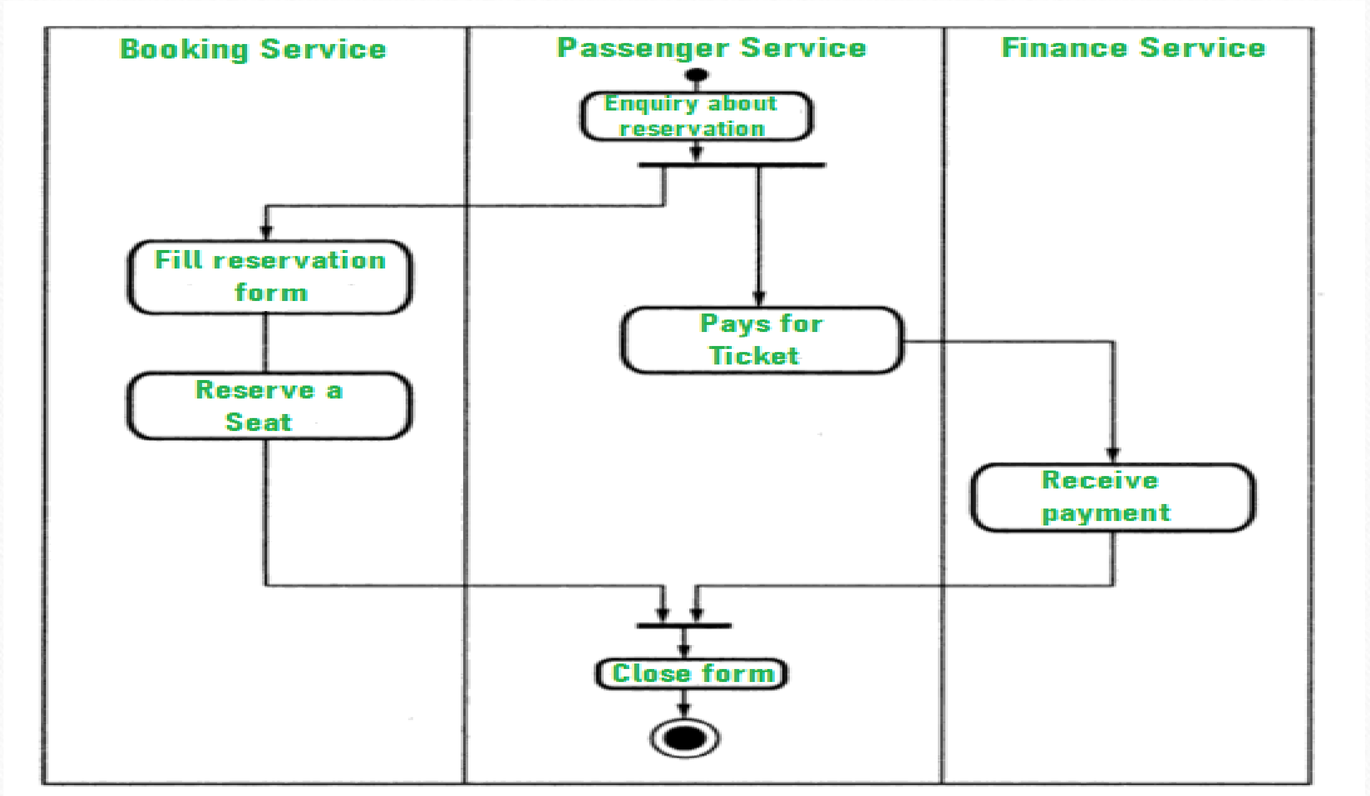
What is a swimlane in Agile?

- **Swimlane Diagram :**
  It is also a graphical representation of the System. Swimlane diagrams are also known as the Rummler-Brache diagram or a cross-functional diagram. Swimlanes are sometimes called functional bands. It simply describes who is responsible for the activities being performed in the activity diagram and how they are responsible. The activity diagram only represents the activities being performed, but Swimlane describes who does what in a process or activity performed.

What is the purpose of a swimlane diagram?

- Swimlane diagrams break down highly complex projects into visually intuitive components that are much easier to analyze when taken separately. In turn, this achieves:
- Better insight into the project and improved communication within teams;
- Fair and optimal workload distribution;
- Easier analysis of project phases and process tweaking on the go.

It is also a graphical representation of the System. Swimlane diagrams are also known as the Rummler-Brache diagram or a cross-functional diagram. Swimlanes are sometimes called functional bands. It simply describes who is responsible for the activities being performed in the activity diagram and how they are responsible. The activity diagram only represents the activities being performed, but Swimlane describes who does what in a process or activity performed.



Swimlane Diagram for Reserving a Ticket

What is a Minimum Viable Product?

- The term Minimum Viable Product or MVP comes from Lean Startup, a methodology that focuses on establishing businesses and products within a short development lifecycle. This MVP quickly and efficiently determines if a proposal is viable without significant cost or risk.
- MVP, is a product with enough features to attract early-adopter customers and validate a product idea early in the product development cycle. In industries such as software, the MVP can help the product team receive user feedback as quickly as possible to iterate and improve the product.

What is the Purpose of a Minimum Viable Product?

- Eric Ries, who introduced the concept of the minimum viable product

A company might choose to develop and release a minimum viable product because its product team wants to:

- Release a product to the market as quickly as possible
- Test an idea with real users before committing a large budget to the product's full development
- Learn what resonates with the company's target market and what doesn't
- In addition to allowing your company to validate an idea for a product without building the entire product, an MVP can also help minimize the time and resources you might otherwise commit to building a product that won't succeed.

Difference between Traditional and Agile Software Development

1.	**It is used to develop simple software.**	**It is used to develop complicated software.**
2.	In this methodology, testing is done once the development phase is completed.	In this methodology, testing and development processes are performed concurrently.
3.	It follows a linear organization structure.	It follows an iterative organizational structure.
4.	It provides less security.	It provides high security.

Difference between Traditional and Agile Software Development

5.	Client involvement is less as compared to Agile development.	Client involvement is high as compared to traditional software development.
6.	It provides less functionality in the software.	It provides all the functionality needed by the users.
7.	It supports a fixed development model.	It supports a changeable development model.
8.	It is used by freshers.	It is used by professionals.
9.	Development cost is less using this methodology.	Development cost is high using this methodology.
10.	It majorly consists of five phases.	It consists of only three phases.

Difference between Traditional and Agile Software Development

11.	**It is less used by software development firms.**	**It is normally used by software development firms.**
12.	Expectation is favored in the traditional model.	Adaptability is favored in the agile methodology.
13.	Traditional software development approaches are formal in terms of communication with customers.	Agile software development methodologies are casual. In other words, customers who work with companies that utilize Agile software development approaches are more likely to interact with them than customers who work with companies that use traditional software development methodology.

Difference between Traditional and Agile Software Development

14.	For starters, typical software development approaches employ a predictive approach. There is full specification and prediction of the software development processes because the product is produced through rigorous and explicit planning. Changes are not permitted in this technique because the time and cost of project development are fixed.	Here, a flexible approach is used as the software development approaches are founded on the notion of continual design improvement and testing relies on team and client feedback.
15.	Examples- • Office productivity suites • Data management software • Media players • Security programs	Examples- • Sky • Phillips • JP Morgan Chase

Difference between Traditional and Agile Software Development

16.	Models based on Traditional Software Development- • Spiral Model • Waterfall Model • V Model • Incremental Model	Models based on Agile Software Development- • Scrum • Extreme Programming (XP) • Crystal • Dynamic Systems Development Method (DSDM) • Feature Driven Development (FDD) • Adaptive Software Development (ASD)

Version and Version Control

As a project progresses, many versions of individual work products will be created The repository must be able to save all of these versions to enable effective management of product releases and to permit developers to go back to previous . The repository must be able to control a wide variety of object types, including text, versions during testing and debugging.

What is a "version control system"?

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

Why Version Control system is so Important?

As we know that a software product is developed in collaboration by a group of developers they might be located at different locations and each one of them contributes to some specific kind of functionality/features. So in order to contribute to the product, they made modifications to the source code(either by adding or removing)..

A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what changes have been made
A separate branch is created for every contributor who made the changes and the changes aren't merged into the original source code unless all are analyzed as soon as the changes are green signaled they merged to the main source code. It not only keeps source code organized but also improves productivity by making the development process smooth.
identified as follows:
1.A release number (and a revision letters if in drafts).
2. The original draft shall be 0.A
subsequent drafts shall be Version 0.B, Version O.C etc.
3. The accepted and issued document is Version 1.0 subsequent changes in draft form become Version 1.0A, 1.0B etc. 4. The accepted and issued second version becomes Version 1.1 or Version 2.0, as determined by the author based on the magnitude of the changes

Release

Software Release Process :
Software release process is also known as Release Management Cycle in which it performs a cycle for developing, testing, deploying, supporting new versions of software basically for Agile development.
It includes modifying, managing, organizing, programming, regulating the software application of different platforms. This process enhances the standard, speed, productivity for developing new software with the updated version.

phases of a software release process

1.Define specific requirements for the release –
If any developer wants to add some changes within the existing software, then it should run in any environment that application should be available for computer application as well as mobile application.
2.Specify your acceptance criteria –
In this all the requirement criteria for the updated version of software application should be at one go for IOS version as well as Android version.
3.Test your software in production –
For new software some test required for the real time running and to check that method of testing dark launching method is used without encoding their methods.

4. **Iterate and refine your product –**
After testing if that application needs some changes before disclosing about the updated version that should be refined first then it should be launched.
5. Release your product to end-users –
After clearing the dark launching tests it allows the product manager to announce the updated version of the software application and to release the codes and it increases the productivity of the developer by a huge method.