Waterfall Model

Waterfall Model

- Was the first process model to be introduced.
- Also referred to as a **linear-sequential life cycle model.**
- Each phase must be completed before the next phase can begin.
- Used in situations where the project requirements are well-understood, stable.

Examples of scenarios where the waterfall model may be used

Construction Projects:

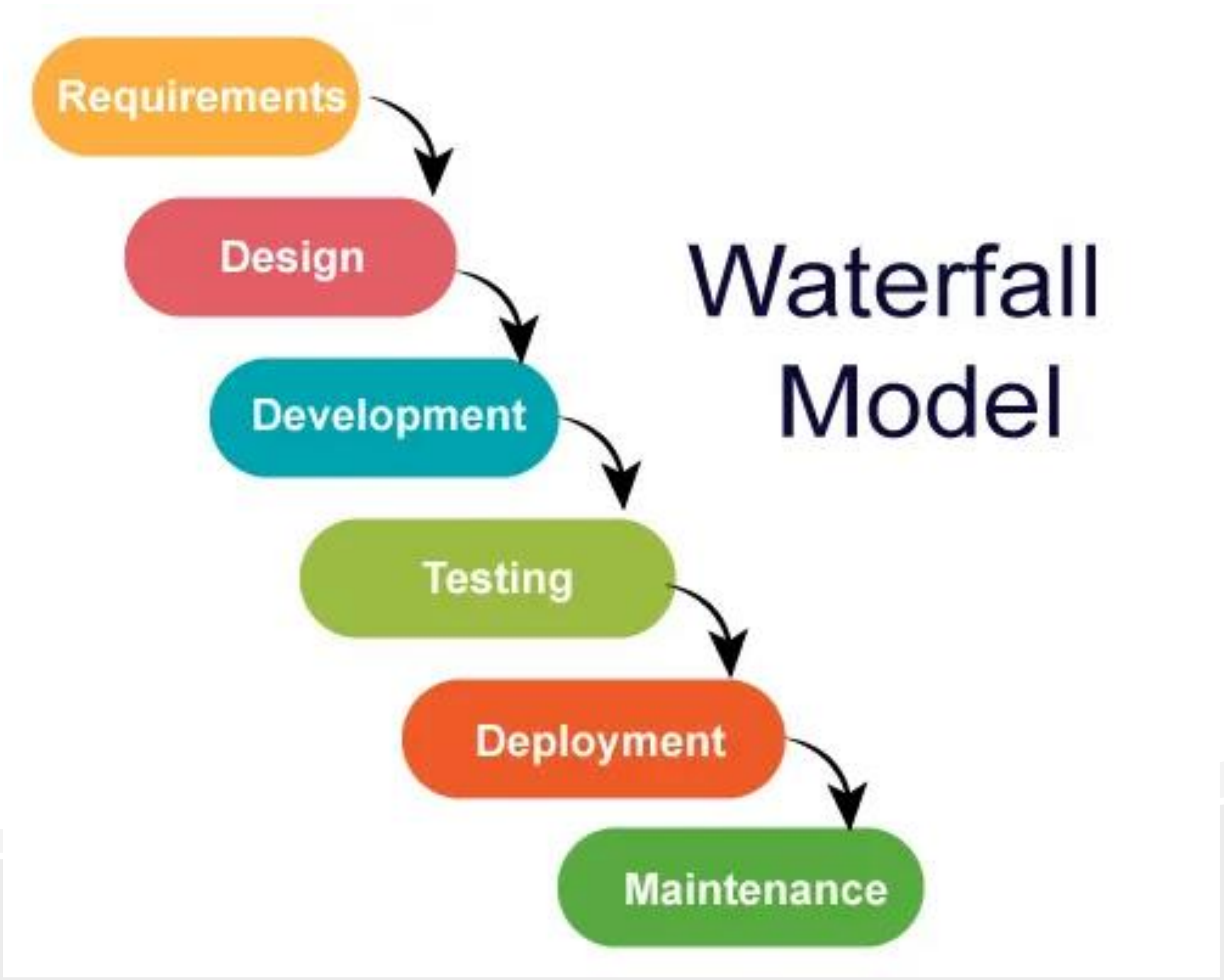
Waterfall can be applied to construction projects, such as building a bridge or a physical structure, where the requirements are well-documented and the sequence of tasks must be followed in a linear manner.

Simple Software Projects:

- In cases where the software requirements are straightforward, clear, and unlikely to change significantly, the Waterfall Model can still be used for software development.
- For example, developing a basic website or a simple mobile app with well-understood features.

- Waterfall Model is less suitable for software development projects in dynamic and rapidly changing environments, where customer needs are evolving, or where there is a high level of uncertainty.
- In such cases, agile methodologies like Scrum or Kanban are more commonly used to accommodate changes and deliver value to users in shorter iterations.

Different phases of the Waterfall Model



Requirements Gathering

- Project team works with stakeholders to gather and document all project requirements.
- Aims to establish a clear understanding of what the software should accomplish.

System Design

- Helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- This includes defining data structures, software components, algorithms, and overall system architecture.

Implementation (Coding)

- Involves coding the software based on the design specifications.
- Also includes unit testing to ensure that each component of the software is working as expected.

Testing

- Software is tested as a whole to ensure that it meets the requirements and is free from defects.

Deployment of system

- Once the software has passed testing and quality assurance, it is deployed to the production environment or released into the market.

Maintenance

- Involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

Advantages of Waterfall model

- Very simple and easy to understand.
- Phases in the classical waterfall model are processed one at a time.
- Processes, actions, and results are very well documented.

Disadvantages of Waterfall model

- Difficult to accommodate change requests
- Not suitable for more significant and complex projects.
- Longer project delivery times, as each phase must be completed before moving on to the next.

Agility – Agile Process

- Agility typically refers to the ability of a development team or organization to respond quickly and effectively to changing requirements and customer needs

Agile Processes

- Set of practices and principles used in software development and project management
- That prioritize flexibility, collaboration, and responsiveness to change.

Some of the most well-known agile processes

- Scrum
- Kanban
- Extreme Programming (XP)

Agile Software Development process typically consists of the following steps:

- **Requirements Gathering**: The customer's requirements for the software are gathered and prioritized.
- **Planning:** The development team creates a plan for delivering the software, including the features that will be delivered in each iteration.
- **Development:** The development team works to build the software, using frequent and rapid iterations.

- **Testing:** The software is thoroughly tested to ensure that it meets the customer’s requirements and is of high quality.
- **Deployment:** The software is deployed and put into use.
- **Maintenance**: The software is maintained to ensure that it continues to meet the customer’s needs and expectations.

Agile Software Development

- Is well-suited to changing requirements and the fast pace of software development.
- Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.

Advantages of agile software development

- **Flexibility:** Agile lets you easily change the project as needed.
- **Customer-Focused**: Agile makes sure the customer is happy with the software.
- **Quicker Releases:** Agile helps you release useful parts of the software faster.
- **Better Quality:** Agile finds and fixes problems early for a better product.

- **Teamwork:** Agile encourages team members to work together and communicate well.
- **Less Risk**: Agile helps identify and solve issues early, reducing the chance of big problems.
- **Transparency**: Everyone knows how the project is going, so no one is surprised.
- **Satisfied Customers:** Agile helps deliver what customers want, making them happy.
- **Easy Changes:** You can make changes to your project without causing big problems.

Disadvantages of agile software development

- **Complexity**: Agile can be hard to manage for very large or complex projects.
- **Lack of Documentation:** Agile may have less documentation, which can make it hard to understand the project's history.
- **Customer Availability:** You need customers to be available for feedback, which may not always be possible.
- **Team Skill Requirement:** Agile needs skilled team members who understand the process.

Extreme Programming

Extreme Programming

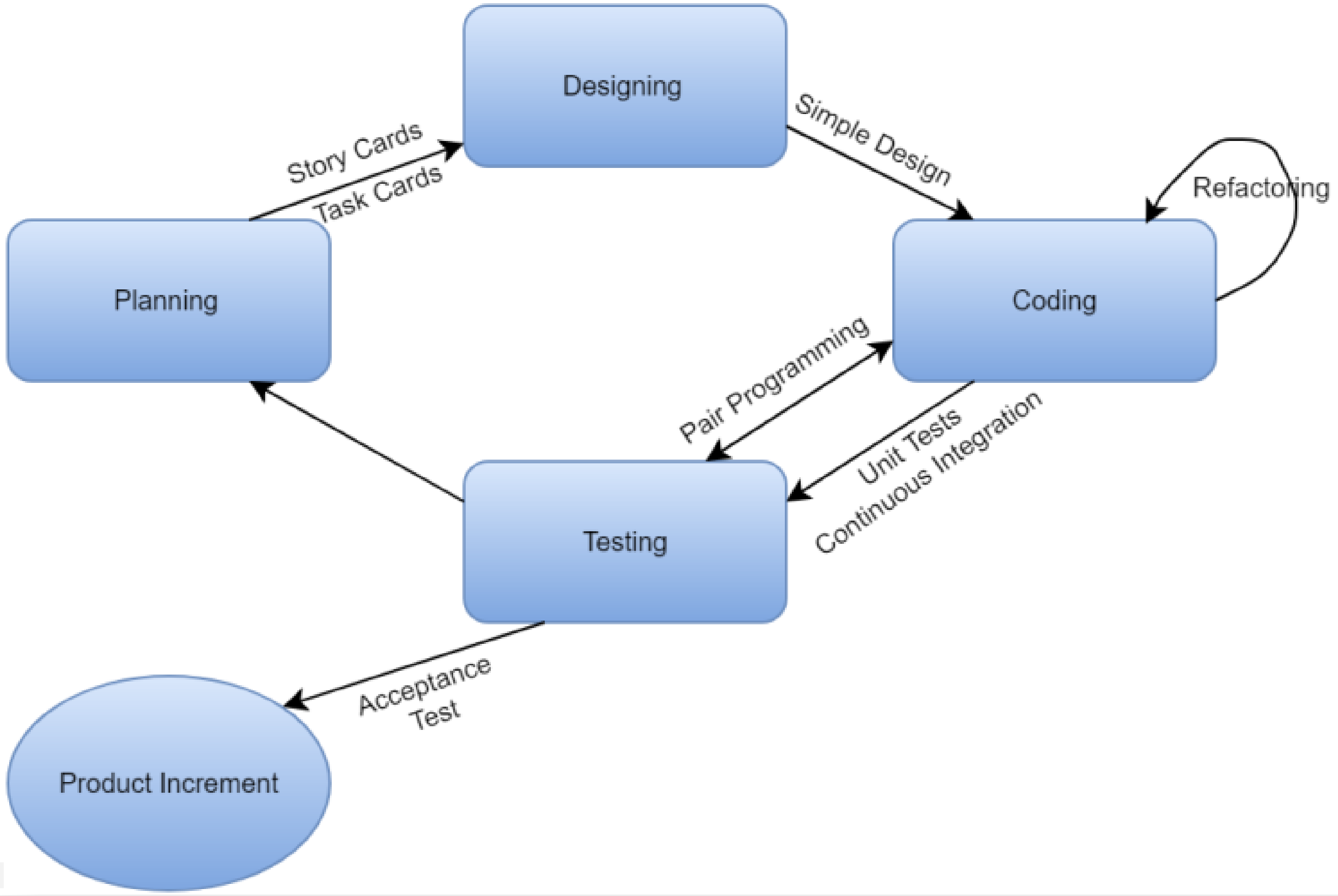
- Used to improve software quality and responsiveness to customer requirements.

- **Customer Collaboration**: XP involves working closely with customers to understand what they want and making changes based on their feedback.

- **Frequent Small Releases:** Instead of waiting a long time to finish everything, XP delivers small parts of the software often, so customers get useful features sooner.

- **Planning Together**: Customers and developers plan the work together, deciding what's most important and how long it will take.

- **Pair Programming:** Two people work together on the same computer to write better code and catch mistakes faster.
- **Continuous Integration:** Developers regularly combine their work, test it, and fix any issues right away.
- **Shared Responsibility:** Everyone on the team is responsible for the code and helps improve it.
- **Customer Availability**: Having the customer available to answer questions and give feedback is very important.


Designing
Story Cards
Task Cards
Simple Design
Refactoring
Planning
Coding
Pair Programming
Unit Tests
Continuous Integration
Testing
Acceptance Test
Product Increment

Advantages of XP

- Saves cost and time required for project realization.
- Developers end up creating simple code that can be improved at any moment.
- Regular testing at the development stage leads to faster creation of the software.

Disadvantages of XP

- Requires everyone on the team to be physically available. XP is not a good option if programmers are separated geographically.
- XP does not measure code quality assurance.
- There is a lack of documentation and defect documentation is not always good. This may lead to the occurrence of similar bugs in the future.

Adaptive Software Development

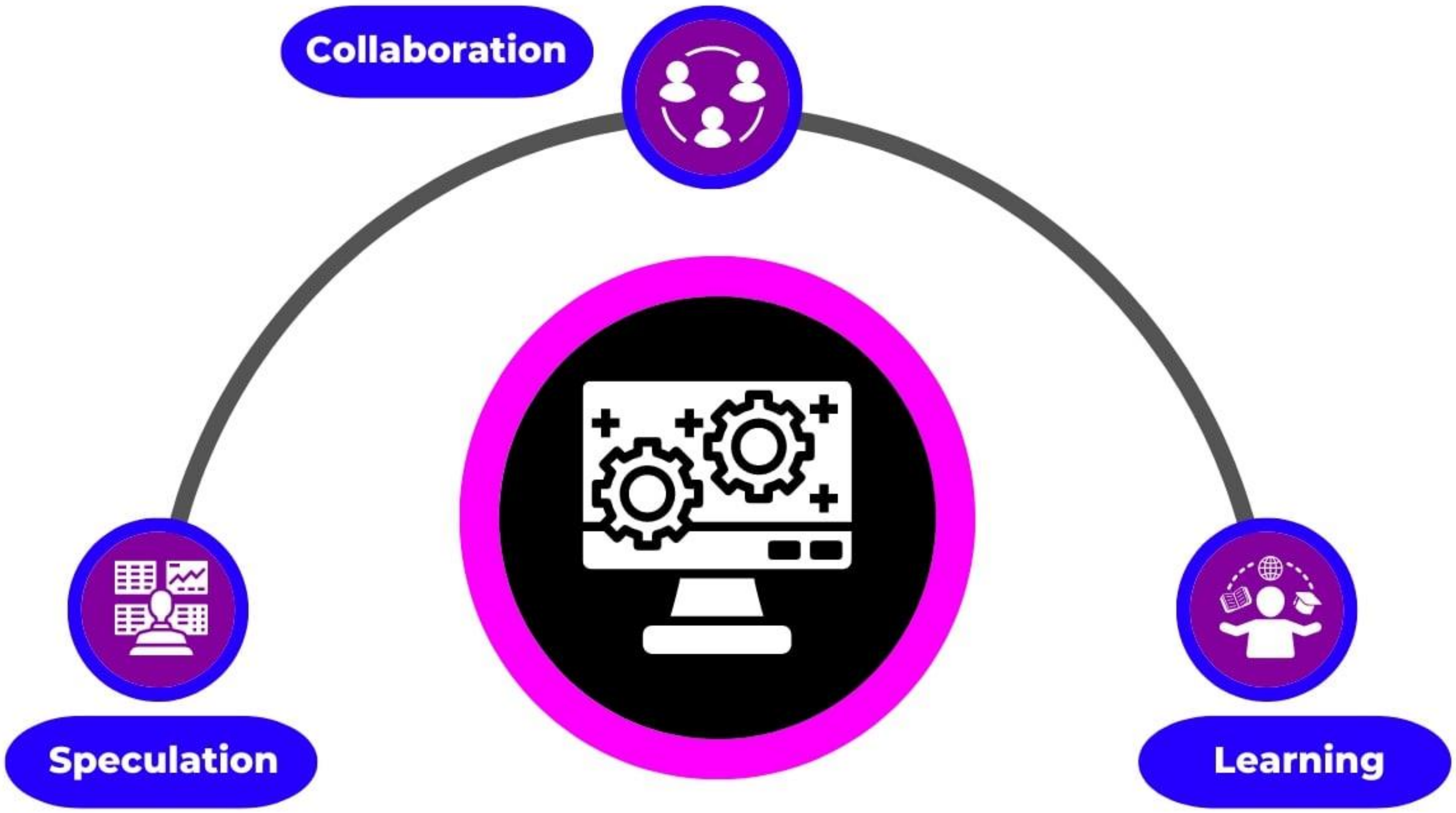
Adaptive Software Development

- Focuses on flexibility and adaptability in response to changing requirements and customer needs.

The framework has three development lifecycle phases

- Speculation
- Collaboration
- Learning

ASD Life- Cycle



Speculate

- In this phase, the focus is on understanding the overall goals and requirements of the project.
- It involves brainstorming, discussing ideas, and gathering initial information.
- It's like laying the foundation for a building; you're figuring out what you need and how it should look.

Collaborate

- During this phase, the development team works closely with customers and stakeholders to refine the project's requirements and objectives.
- Continuous collaboration and feedback are key.
- Collaboration is the skill of working together with different members having different expertise to produce the best outcomes.
- It's like architects and builders working together to make sure the building meets the needs of the people who will use it.

Learn

- In the "Learn" phase, the team takes the knowledge gained from the previous phases and uses it to develop the software incrementally.
- It involves regular testing, feedback, and adjustments to meet changing needs.
- This phase is like constructing the building, but with the ability to make changes as you go along based on what you've learned.

Example

Scenario: Building a Mobile App for a Coffee Shop

1. **Speculate Phase:**

- **Goal**: The coffee shop wants to create a mobile app to allow customers to order coffee and pastries for pickup.
- **Activities:** In this phase, the coffee shop owner, along with a software development team, discusses the high-level goals. They gather initial ideas and requirements, such as the types of drinks and pastries offered, order placement process, and basic design concepts.

2. Collaborate Phase:

Goal: Refine the app's features and design based on customer feedback and business needs.

Activities:

- The coffee shop owner and the development team collaborate closely.
- They talk to customers to understand their preferences.
- Customers provide feedback on the app's initial design and functionality, suggesting improvements.
- The team adapts the app based on this feedback, making changes to the menu, order process, and design.

3. Learn Phase:

Goal: Develop and improve the app iteratively, incorporating lessons learned from user interactions.

Activities:

The team starts building the app with the refined requirements.

They release an initial version for a small group of customers to use.

As people use the app, the team collects data on how it's being used and gathers more feedback.

They make regular updates to enhance the app, adding new features, fixing bugs, and adjusting the user interface.

Advantages of Adaptive Software Development (ASD):

- **Flexibility**: ASD allows you to easily adapt to changes in project requirements, which is helpful when you're not sure exactly what the final product should look like.
- **Customer Satisfaction**: By involving customers and continuously getting their feedback, ASD increases the chances of delivering a product that meets their needs.
- **Risk Reduction**: ASD helps identify and address issues early, reducing the chances of major problems later in the project.
- **Higher Quality**: With a focus on testing and collaboration, ASD aims for a higher-quality end product.

Disadvantages of Adaptive Software Development (ASD):

- **Uncertainty:** The flexible nature of ASD can be challenging if you have strict deadlines or if customers frequently change their minds.
- **Complexity:** The iterative approach can make the project management process more complex compared to traditional methods.
- **Documentation:** ASD often relies on minimal documentation, which can be a disadvantage if thorough documentation is essential for your project.
- **Resource Intensive**: Frequent collaboration and adjustments may require more time and resources than initially planned.

Dynamic Systems Development Method (DSDM)

- Is a software development approach that focuses on delivering high-quality systems quickly by involving users and stakeholders throughout the process.

Feature Driven Development

- Building software one feature at a time.
- Breaks down the development process into manageable pieces, with each piece representing a specific feature or functionality of the software

- FDD is like constructing a building one room at a time.
- Instead of tackling the entire project all at once, you focus on building individual rooms, ensuring each one is completed and functional before moving on to the next.

Key aspects of Feature Driven Development (FDD) include:

- **Domain Object Modeling**: Creating a detailed model of the problem domain to understand the business or functional requirements.
- **Feature List:** Compiling a comprehensive list of features that the software should have, often based on user needs and the domain model.
- **Feature-by-Feature Development:** Developing and testing one feature at a time, with each feature going through a series of design, implementation, and testing steps.

- **Regular Inspections**: Conducting inspections or reviews to ensure that each feature meets its intended requirements and is of high quality.
- **Team Collaboration:** Promoting collaboration among team members, including designers, developers, and testers, to work together efficiently.
- **Detailed Design:** Creating detailed designs for each feature before implementing them.
- **Regular Builds:** Continuously building and testing the software to verify that features are working as expected.

Example: Creating a Mobile Phone Contact App

- Imagine you're developing a basic mobile phone contact app, like the one you use to store and manage phone numbers and names. Here's how you might apply FDD:

- **Domain Object Modeling**: You start by thinking about the fundamental components of your contact app. This includes defining objects like "Contact," "Phone Number," and "Name."

Feature List:

- **Feature 1**: **Adding Contacts:** The first feature you work on is the ability to add contacts. You design the "Add Contact" screen, implement the functionality to input names and phone numbers, and make sure contacts can be saved.

- **Feature 2: Viewing Contacts**: Once adding contacts is complete, you move on to the feature of viewing contacts. You design the "View Contacts" screen, implement the display of saved contacts, and make sure you can scroll through the list.

- **Feature 3: Editing Contacts**: With viewing in place, you work on the feature of editing contacts. You design the "Edit Contact" screen, implement the ability to modify contact details, and ensure changes are saved.

- **Feature 4**: **Deleting Contacts**: The last feature in this example is deleting contacts. You create a "Delete Contact" feature, implement the functionality to remove contacts, and confirm that deleted contacts are removed from the list.

- **Regular Inspections**: After completing each feature, the team conducts inspections and testing to review the feature's functionality, usability, and quality. Any issues or improvements are addressed before moving to the next feature.

- **Team Collaboration:** Throughout the development process, team members, including designers, developers, and testers, collaborate closely. They coordinate their efforts to ensure that each feature is integrated seamlessly into the email application.

- **Regular Builds**: Frequent builds of the application are generated and tested to verify that each feature is working as expected. This allows for early detection and resolution of any issues.

Lean Software Development

Lean Software Development

- It focuses on maximizing customer value while minimizing waste and unnecessary processes in software development

Key Principles of Lean Software Development

- **Eliminate Waste**: The core principle is to eliminate any activity or process that does not add value to the customer. This includes avoiding unnecessary documentation, redundant tasks, and features that aren't essential.

- **Amplify Learning:** Lean encourages continuous learning and adaptation. Developers should learn from feedback and make improvements throughout the development process.

- **Decide as Late as Possible:** Keep important decisions, especially those related to features and design, until you have the most information and can make better choices.

- **Deliver as Fast as Possible:** Aim to deliver small, valuable increments of the software as quickly as possible to get feedback and provide value to users early on.

- **Empower the Team:** Give the development team the autonomy and responsibility to make decisions and solve problems. Trust the expertise of the people doing the work.
- **Build Integrity In**: Focus on building quality into the product from the beginning rather than fixing defects later. This involves rigorous testing and continuous integration.
- **Optimize the Whole:** Look at the entire software development process, not just individual steps, and find ways to make it more efficient. This can involve streamlining communication, reducing handoffs, and improving collaboration.

Example: Building a To-Do List App

Imagine you're tasked with creating a simple to-do list app using Lean Software Development principles:

Eliminate Waste:

- Minimal Features: You start by designing the app with only the essential features—adding tasks, marking them as complete, and deleting them.
- You avoid adding unnecessary features like color customization or complex sorting.

Amplify Learning:

- **User Feedback:** After releasing the initial version, you gather feedback from users. They mention that they'd like the ability to set due dates for tasks.
- **Continuous Improvement:** Based on this feedback, you quickly implement due dates in the next app update, ensuring that users find the app more valuable

Decide as Late as Possible:

Design Choices: You defer design choices like color schemes and fonts until you've received user feedback about their preferences.

Deliver as Fast as Possible:

Minimum Viable Product: You release a basic version of the app with core features (add, mark, delete tasks) as soon as possible. Users can start using it while you work on additional features.

Empower the Team:

- Autonomous Development: You, as the developer, have the authority to make decisions about technical aspects and feature implementations without excessive oversight.

Build Integrity In:

- Automated Testing: You use automated tests to check that new updates and features do not introduce bugs or break existing functionality.

Optimize the Whole:

- Efficient Workflow: You establish an efficient workflow for app development, ensuring smooth collaboration between design, development, and testing.

Tool set for Agile Process.

Project Management Tools:

- **Jira**: Widely used for Scrum and Kanban, Jira helps plan, track, and manage tasks and user stories. It offers features for backlog management, sprint planning, and issue tracking.
- **Trello:** Known for its simplicity, Trello is a visual project management tool using Kanban boards. Teams can create boards for different projects and move cards (tasks) through various stages.
- **Asana:** A versatile project management tool with task lists, boards, and timelines. It's suitable for various Agile methodologies.

Collaboration and Communication Tools

- **Slack**: A real-time messaging platform that facilitates team communication, collaboration, and file sharing. Slack helps Agile teams stay connected.
- **Microsoft Teams**: Integrated with Office 365, Microsoft Teams offers chat, video meetings, file storage, and app integration to streamline team collaboration.

Version Control Tools:

- **Git**: A distributed version control system used for tracking changes in source code. Git, along with platforms like GitHub and GitLab, enables collaboration among developers.

Continuous Integration and Continuous Deployment (CI/CD) Tools

- **Jenkins**: An open-source automation server for building, testing, and deploying code changes continuously. Jenkins integrates with various development tools and platforms.

- **Travis CI:** A cloud-based CI/CD service that automates building, testing, and deploying applications from repositories like GitHub.

Test Management Tools:

- **TestRail**: A test management tool for creating test cases, managing test runs, and tracking test results. It integrates with various issue-tracking and development tools.
- **JIRA Test Management**: This is a plugin for Jira that adds test management capabilities to Jira, making it easier to manage test cases and trace them to user stories.

Scrum

Manoj

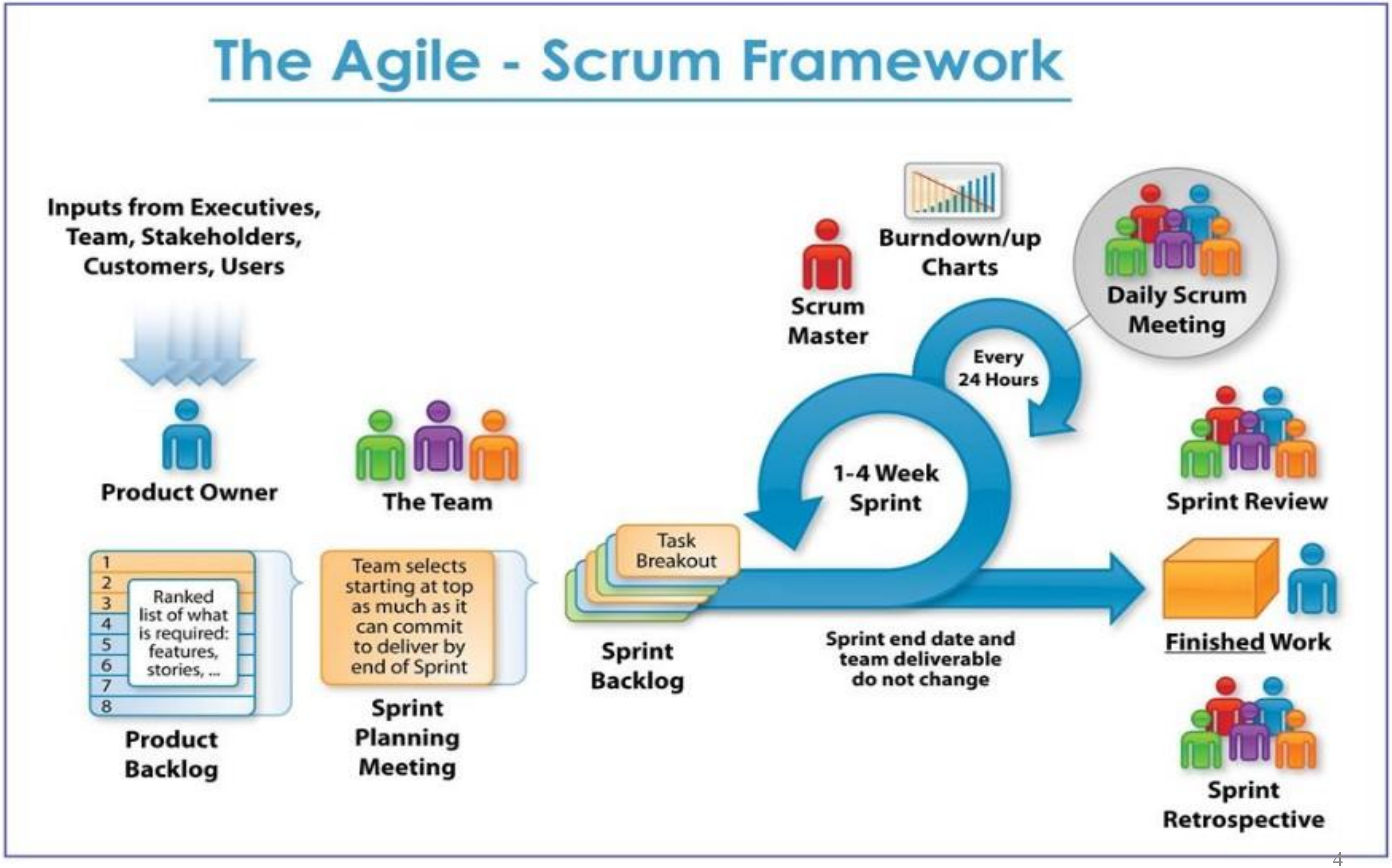
Scrum

- Framework that has been used to manage complex product development
- Within which we can develop software with teamwork.
- Based on agile principles.
- Within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

Scrum Roles and Responsibilities

Three main roles in SCRUM

- SCRUM MASTER
- PRODUCT OWNER
- DEVELOPMENT TEAM


The Agile - Scrum Framework
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
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
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint end date and team deliverable do not change
Sprint Review
Finished Work
Sprint Retrospective

ScrumMaster

- Making the process run smoothly
- Removing obstacles that impact productivity
- Organizing and facilitating the critical meetings
- Keep information about the Team’s progress up to date and visible to all parties.

Product Owner

- Person serves as the communication between the development team and its customers.
- Product owner is responsible for ensuring that expectations for the completed product are communicated and agreed upon.

Development team

- Responsible for producing the product, it must also have the authority to make decisions about how to perform the work.
- Team members decide how to break work into tasks, and how to allocate tasks to individuals, throughout the Sprint.
- The Team size should be kept in the range from five to nine people, if possible.
- (A larger number make communication difficult, while a smaller number leads to low productivity and fragility.)

scrum artifacts

scrum artifacts

- Refers to information that scrum team use to describe a product that’s being developed.
- Process framework
    1. Product Backlog
    2. Sprint Backlog
    3. Burn-Down Chart
    4. Increment

Product Backlog

- Prioritized list of all the features, user stories, and tasks that need to be completed in the project. It's managed by the product owner.
- It’s compiled from input sources like customer support, competitor analysis, market demands, and general business analysis.

Sprint backlog

- Sprint backlogs are created by the development teams to plan deliverables for future increments and detail the work required to create the increment.
- Sprint backlogs are created by selecting a task from the product backlog and breaking that task into smaller, actionable sprint items.

Product increment

- The Increment is the sum of all the Product Backlog items completed during a Sprint combined with the increments of all previous Sprints.
- At the end of a Sprint, the new Increment must be a working product, which means it must be in a useable condition.

Sprint Burn-Down Chart

- Team tracks this total work remaining for every Daily Scrum to project the likelihood of achieving the Sprint Goal.
- By tracking the remaining work throughout the Sprint, the Team can manage its progress.
- Product Owner tracks this total work remaining at least every Sprint Review.
- he Product Owner compares this amount with work remaining at previous Sprint Reviews to assess progress toward completing the projected work by the desired time for the goal.

User Stories

User Stories

- It is a short, written explanation of a particular user’s need and how it can be fulfilled.
- It is written in easily accessible language to provide a clear picture of what the user requires.
- A collection of user stories is referred to as an epic.
- A product owner will be responsible for managing these user stories, but they can be written by any Agile team member.

User stories: Examples

- The chosen user story format will outline the “who,” “what,” and “why” of a particular requirement.

    1. Who wants something?
    2. What do they want?
    3. Why do they want it?

- The following template is one of the most common:

“As [persona], I want to [action], so that I can [benefit].”

- **Example 1: An online gamer**

“As an online gamer, I want to have a multiplayer option so that I can play online with friends.”

- **Example 2: An e-commerce shopper**

“As an e-commerce shopper, I want to filter my searches so I can find products quickly.”

Release planning

Release planning

- Is the creation of a plan that describes how a product will be released into the market.
- Plan should take into account all of the factors that could affect the success of the product launch, including team availability, deadlines, and dependencies.
- By creating a plan that takes into account all the potential variables, developers can ensure that their product is released on time and meets all the necessary requirements.

Daily Scrum

- What did we do yesterday?
- What are we doing today?
- What are the challenges?

sprint review meeting

sprint review meeting

- Is an informal meeting held at the end of a sprint, in which the Scrum team shows what was accomplished during this period.
- This typically takes the form of a demonstration of new features, with the goal of creating transparency and generating feedback.
- Purpose of a Scrum sprint review is not to provide a status update or make a presentation to stakeholders; it is to collect feedback on the actual product

Sprint review meeting agenda

- Reviewing the goal for the sprint
- Demonstrating new features implemented during the sprint
- Requesting feedback from the stakeholders
- Discussing work not yet accomplished
- Identifying risks and impediments
- Reviewing project increment objectives
- Looking ahead to the next sprint, using the top lines from the product backlog

Definition of Done

Definition of Done

- ‘Potentially Shippable’ gives a state of ‘confidence’ that what we developed in the Sprint is actually ‘done’.
- Indicates that the built product doesn’t have any ‘undone work’ and is ready to ship.
- If the Scrum teams have produced potentially shippable product, there must be a well-defined, agreed-upon “Definition of Done”.

Features of Definition of Done

- The code should be peer-reviewed
- Code is Checked-In
- Code is deployed to the test environment
- Testing should be passed by feature/code
- The code should give smoke testing
- Documentation of code
- Help documentation is updated
- The Stakeholders approve the feature.

Planning poker

Planning poker

- Planning poker, also known as “scrum poker” and “pointing poker”
- Is a gamified technique that development teams use to guess the effort of project management tasks.
- These estimations are based on the entire group’s input and consensus, making them more engaging and accurate than other methods

How does planning poker work

Step 1: Hand out the cards to participants

- To start a poker planning session, the product owner or customer reads an agile user story or describes a feature to the estimators.
- For example:
- “Customer logs in to the reservation system”
- “Customer enters search criteria for a hotel reservation”
- Team members of the group make estimates by playing numbered cards face-down to the table without revealing their estimate (Fibonacci values: 1,2,3,5,8,13,20,40)
- Cards are simultaneously displayed
- The estimates are then discussed and high and low estimates are explained
- Repeat as needed until estimates converge

Card(s)	Interpretation
0	Task is already completed.
1/2	The task is tiny.
1, 2, 3	These are used for small tasks.
5, 8, 13	These are used for medium sized tasks.
20, 40	These are used for large tasks.
100	These are used for very large tasks.
<infinity>	The task is huge.
?	No idea how long it takes to complete this task.
<cup of coffee>	I am hungry ☺

Step-01: Organization of the game

- Full team will involve in Planning Poker game.
- Will require 2–4 hours
- Also require a large table where all members will sit.
- Each team member will have a set of cards
- In each card set there will have 10 cards with numbering 0, 1, 2, 3, 5, 8, 13, 21, 40 and 100 i.e. 1st card’s no will 0, 2nd card’s no will 1, 3rd card’s no will 2, 4th card’s no will 3, 5th will 4, 6th will 8, 7th will 13, 8th will 21, 9th will 40 and 10th will 100

13
8
20
?
40
13

Step 2: Read the story out loud

- The moderator (either the product owner or product manager) narrates the story to the group.
- If participants have any questions, the moderator answers them.

Step 3: Discuss the story

- Once the group finishes listening to the story, everyone shares their views on it.
- Some of these discussion points will likely include:

    1. How should we handle the work?
    2. How many people are expected to get involved?
    3. What skills will be needed to work on the story?
    4. How should we tackle any roadblocks that delay progress?

The group will also try to learn more about the story and ask questions to understand it better.

PRODUCT BACKLOG
scrum master

Step-04: Playing Cards

- After discussion team members will ask to point for that story i.e. Product owner will ask them Now Select your card for assign Story Point to this story
- Each team member will select their card and put it on the table with face down, so that others can’t see his/her point.
- Once everyone selected their card the product owner will say Now Show
- As soon as the Product owner say this everyone will turn over their card so that everyone present their can see the estimated number

5
5
5
8
Susan
Vadim
Ann
Chris

Step-05: Achieving Consensus

- Most of the time one thing will identify which is a huge mismatch between given numbers.
- In that case members with highest and lowest numbers will ask why their chose estimates
- They will give some justification and discuss with the team regarding this issue.
- This will also help the team if they have any misunderstanding or vogue idea about the story and adjusting their estimation
- After justification the process i.e. Step-04 will continue again
- This process will continue until all members agrees to a common estimate

- Once all team members agrees to a common estimate then Product owner will record that point and go for next story and will continue from step-02 to step-05
- If after 5–6 rounds of playing the game estimation fail meet all members agreement then put it aside for revisit later

Putting a story aside allows for:

- The Product Owner do more research and provide more clarity.
- The developers to investigate the technical options or details (reducing the technical uncertainty).

Story Points

Story point

- Story point is nothing but a measuring unit.
- In simple terms it’s a number that tells the team how complex and big the story is.
- Story size could be related to complexity, uncertainties etc.

- Say you are going on a vacation from city A to city B which is 295 miles away.
- I ask how much time would it take to reach B. Can you give me the exact estimate in hours considering traffic, tolls, weather conditions and of course the vehicle condition.
- I guess no, because If you say 10 hours then that is inaccurate and a vague estimate.
- Now what if I tell you
- On this route (for past 6 months) average speed for all medium load vehicles is 50 miles/hours
- Gets easy ?
- Now you can tell, it would roughly take around 6 hours. (estimate gets better by 4 hours)
- Similarly in software development, product manager has a feature. She translates the feature into multiple small stories and can easily make out the required time to deliver as per the velocity of the team.

- A story size can be easily evaluated after some practice. During initial stages, it is not easy to estimate story points so we defined some baseline stories for each story pointer.
- We used these baselines stories to estimate story points. Depending upon the tasks involved and complexity of the feature, one can easily compare it with other stories and identify the story points.

Extreme Programming (XP)

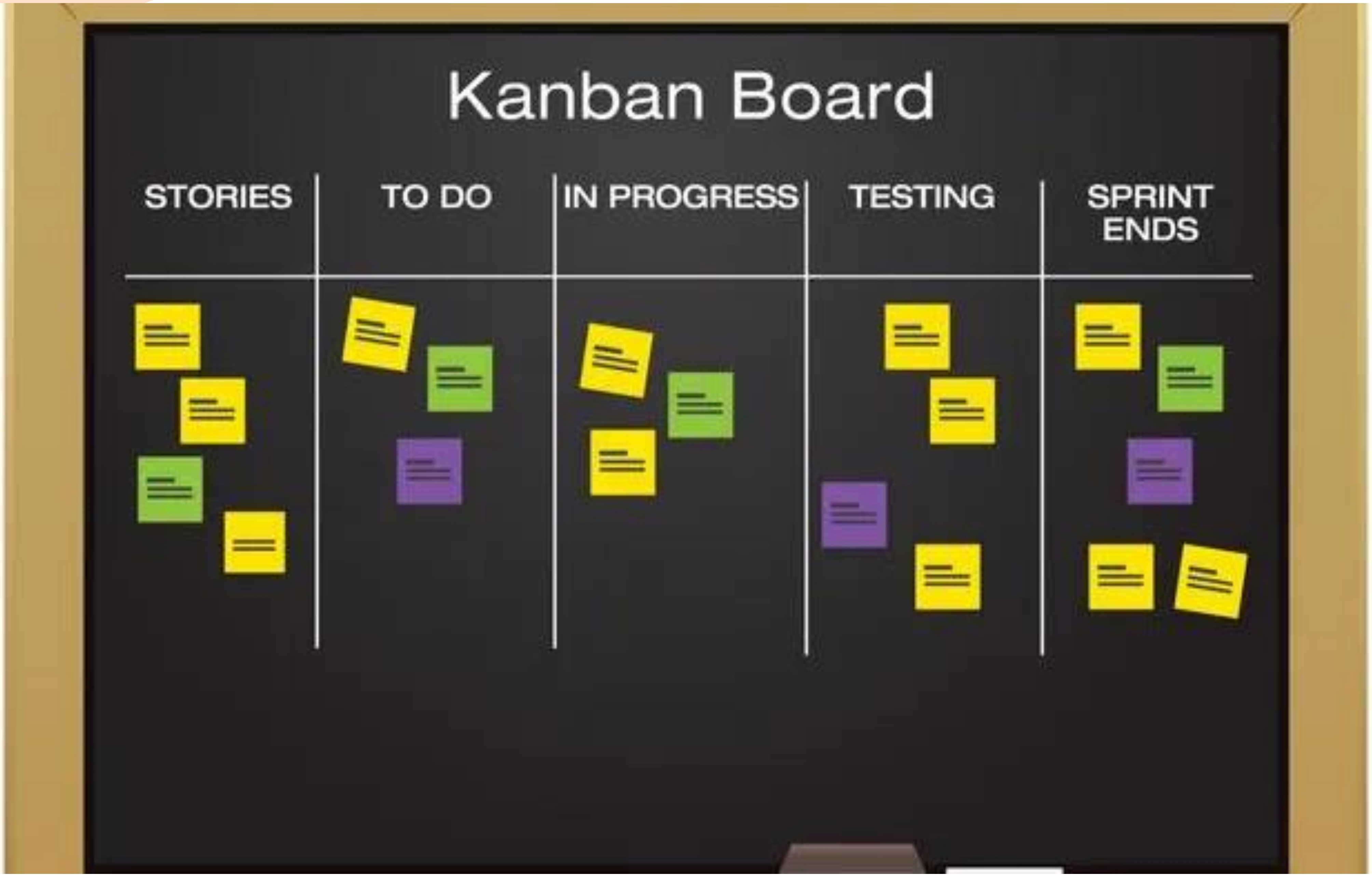
Extreme Programming (XP)

- Is an agile software development framework that aims to produce higher quality software and higher quality of life for the development team.

Kanban

Kanban

- Kanban is all about visualizing our work, limiting work in progress, and maximizing efficiency (or flow).
- Kanban teams focus on reducing the time a project takes (or user story) from start to finish.
- They do this by using a kanban board and continuously improving their flow of work.


Kanban Board
STORIES
TO DO
IN PROGRESS
TESTING
SPRINT ENDS