SDLC_Agile_DevOps_Jira

What is Software Development?

Topics Covered

Software Development

Development Life Cycle

What is Software Development?

Software Development is a process of converting customers ideas into a complete software.

Idea

Development

Software

Software Development

software development life cycle (or SDLC)

a methodology that defines the steps of a software development project

Development Lifecycle

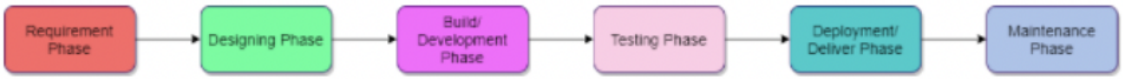
Software Development Lifecycle

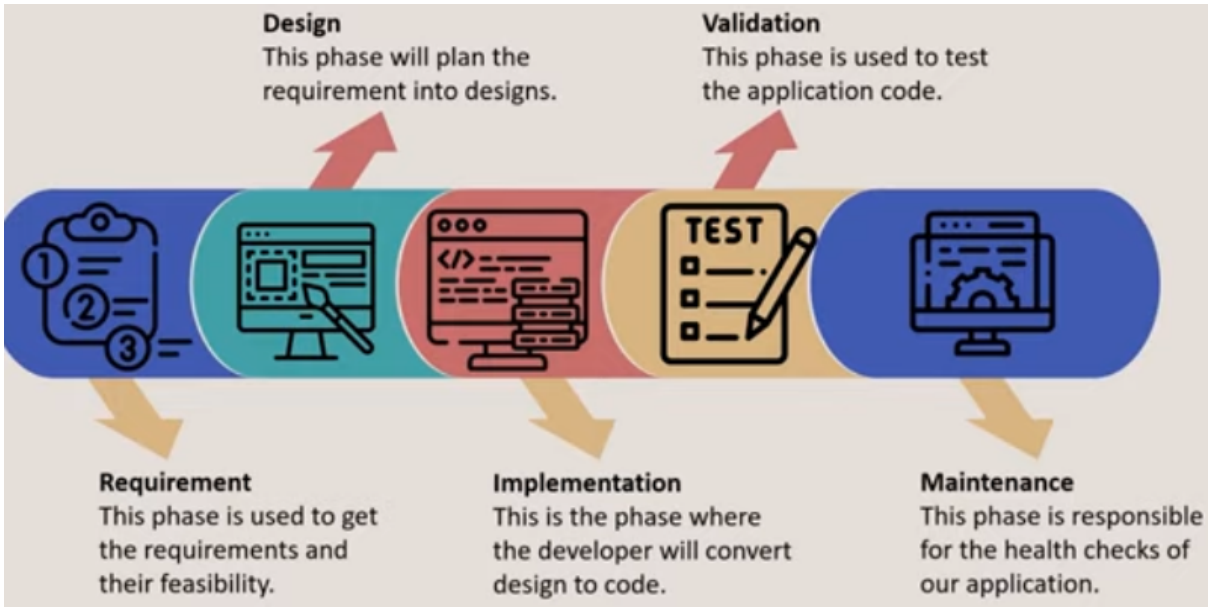
- Software Development lifecycle is a practice used to maintain the quality of software and deliver the product in time
- Lifecycle consists of 5 important phases
- It is also referred to as the Application Development Lifecycle
- This transform the idea into functional and operational architecture

Phases of SDLC

The SDLC process consists essentially of the following phases:

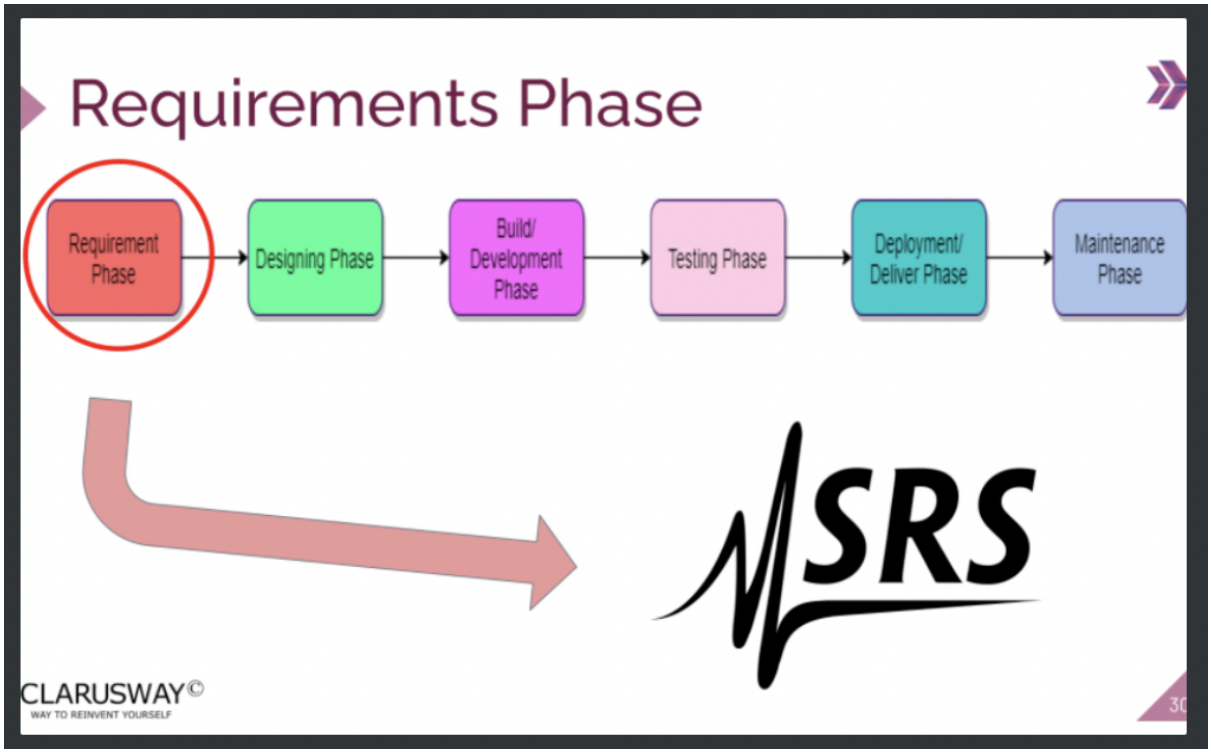
- Requirement Phase
- Design Phase
- Build/Development Phase
- Testing Phase
- Deployment/Deliver Phase
- Maintenance





The requirement is the first and the most critical phase of SDLC for both the developing team and the project manager.

SRS (Software Requirement Specification) "SRS is a detailed description of a software system to be developed with requirements. The SRS is developed based on the agreement between customers and contractors. It may include the use cases of how a user is going to interact with a software system. "



Software Development and Software Testing

No matter which SDLC model is chosen, test activities should start in the early stages of the life cycle, adhering to the testing principle of early testing.

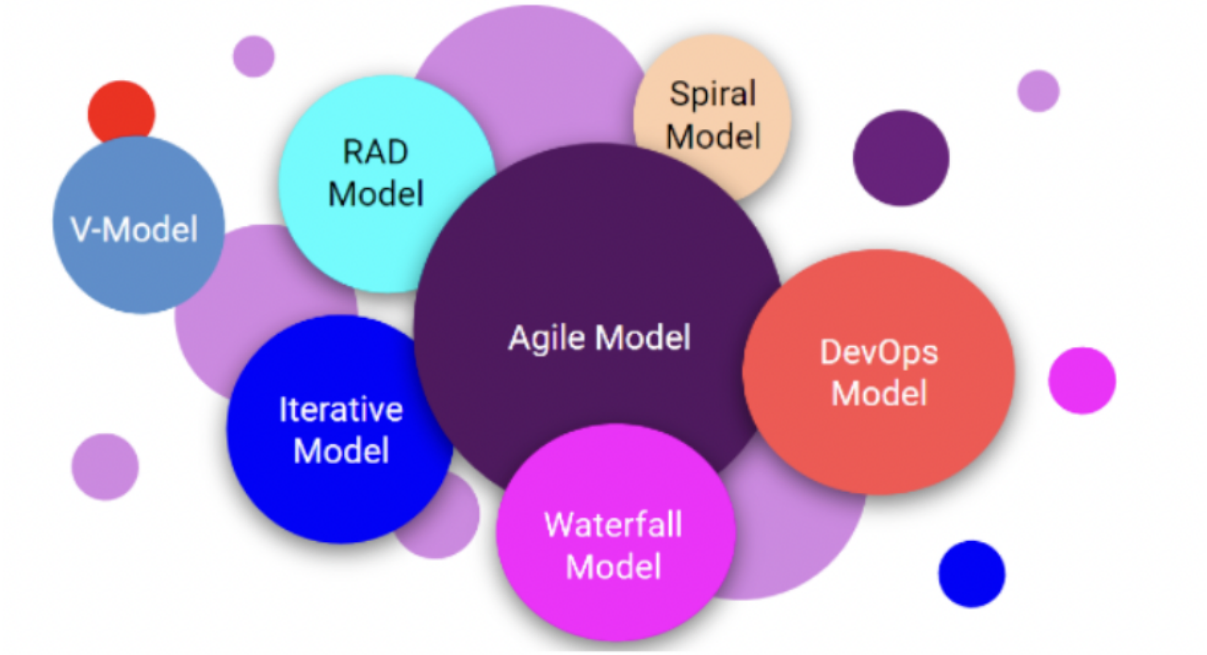
Verification & Validation

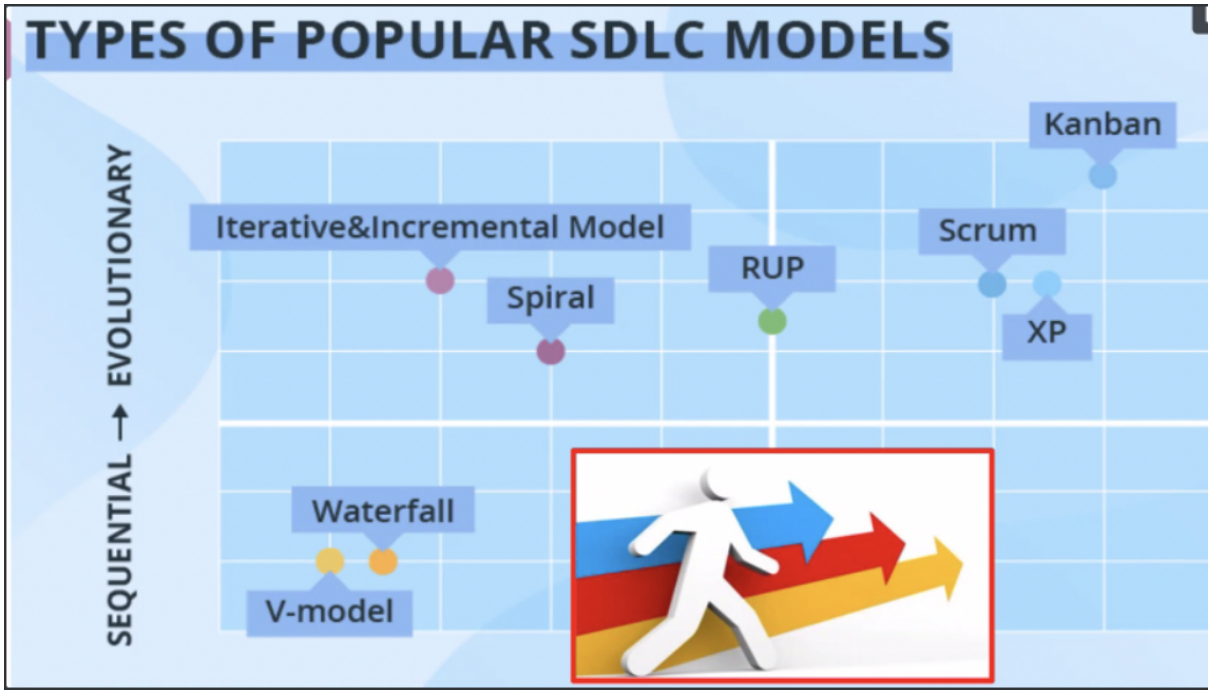
In every development life cycle, a part of testing is focused on **verification** testing, and another part is focused on **validation** testing.

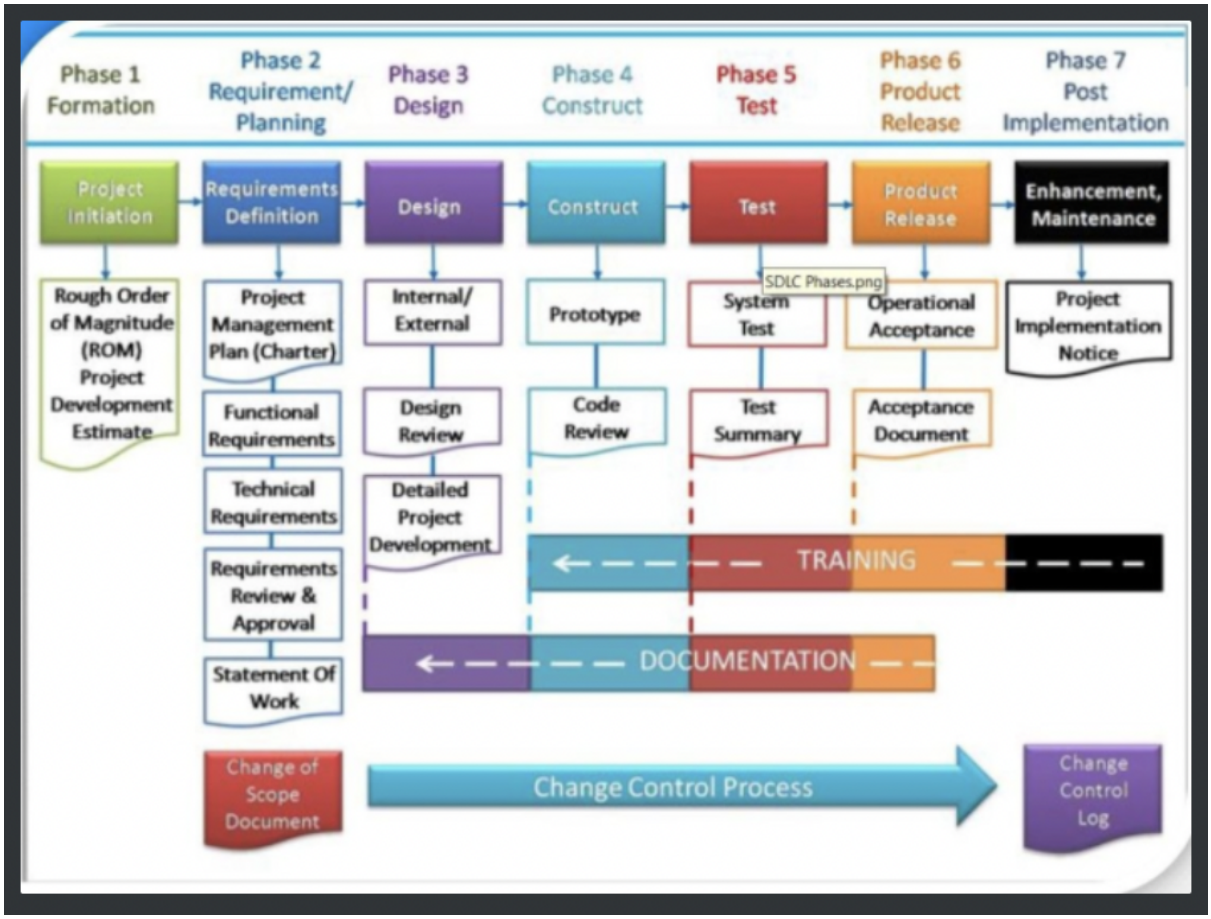
Verification is concerned with evaluating a work product, component, or system to determine whether it meets the requirements set. Verification focuses on the question **Is the deliverable built according to the specification?**.

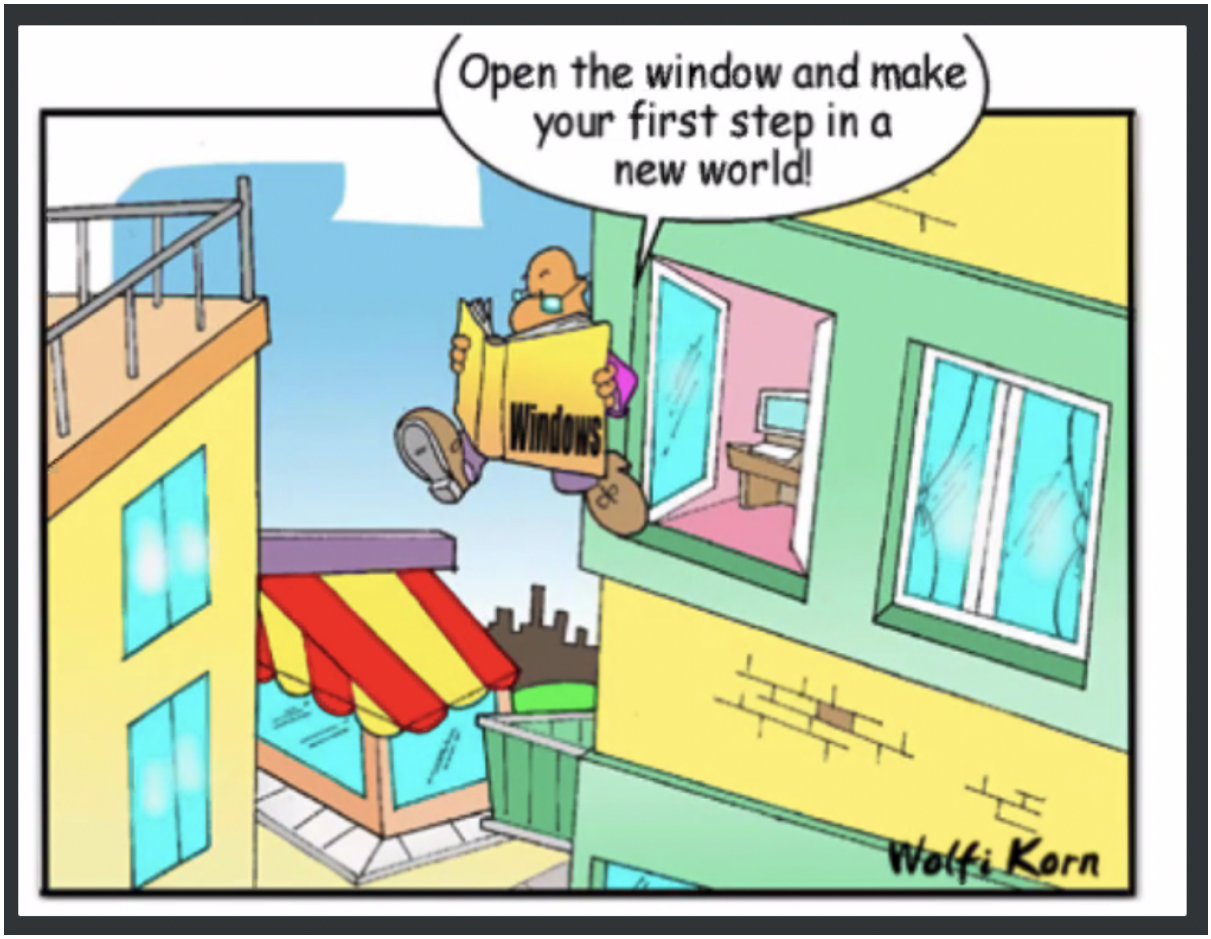
Validation is concerned with evaluating a work product, component, or system to determine whether it meets the user needs and requirements. Validation focuses on the question **Is the deliverable fit for purpose, and does it provide a solution to the problem?**.

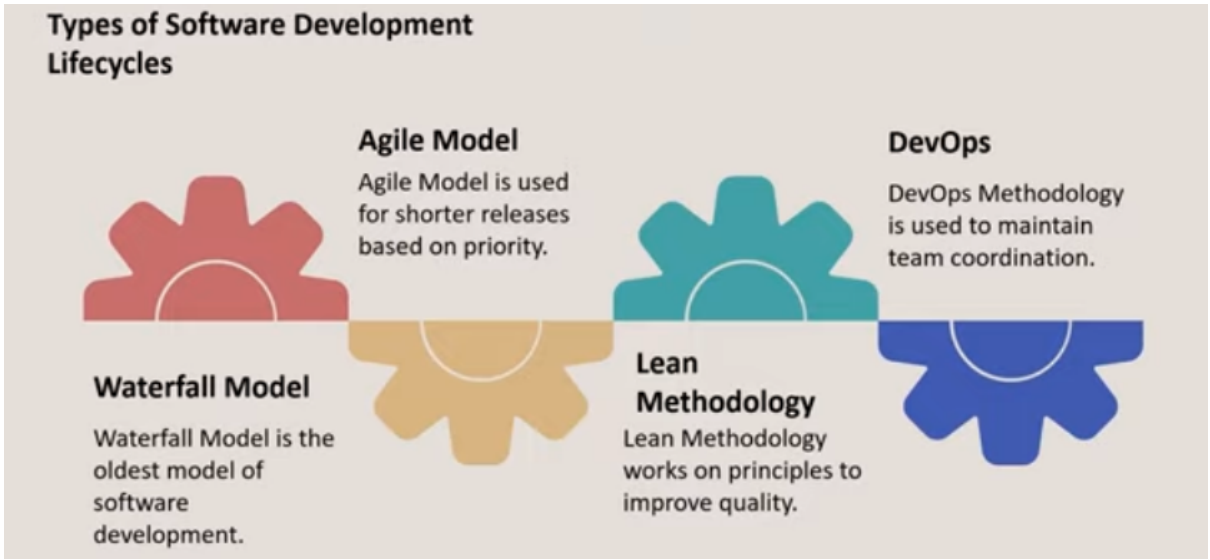
There are various Software development models or methodologies. They are as shown below:



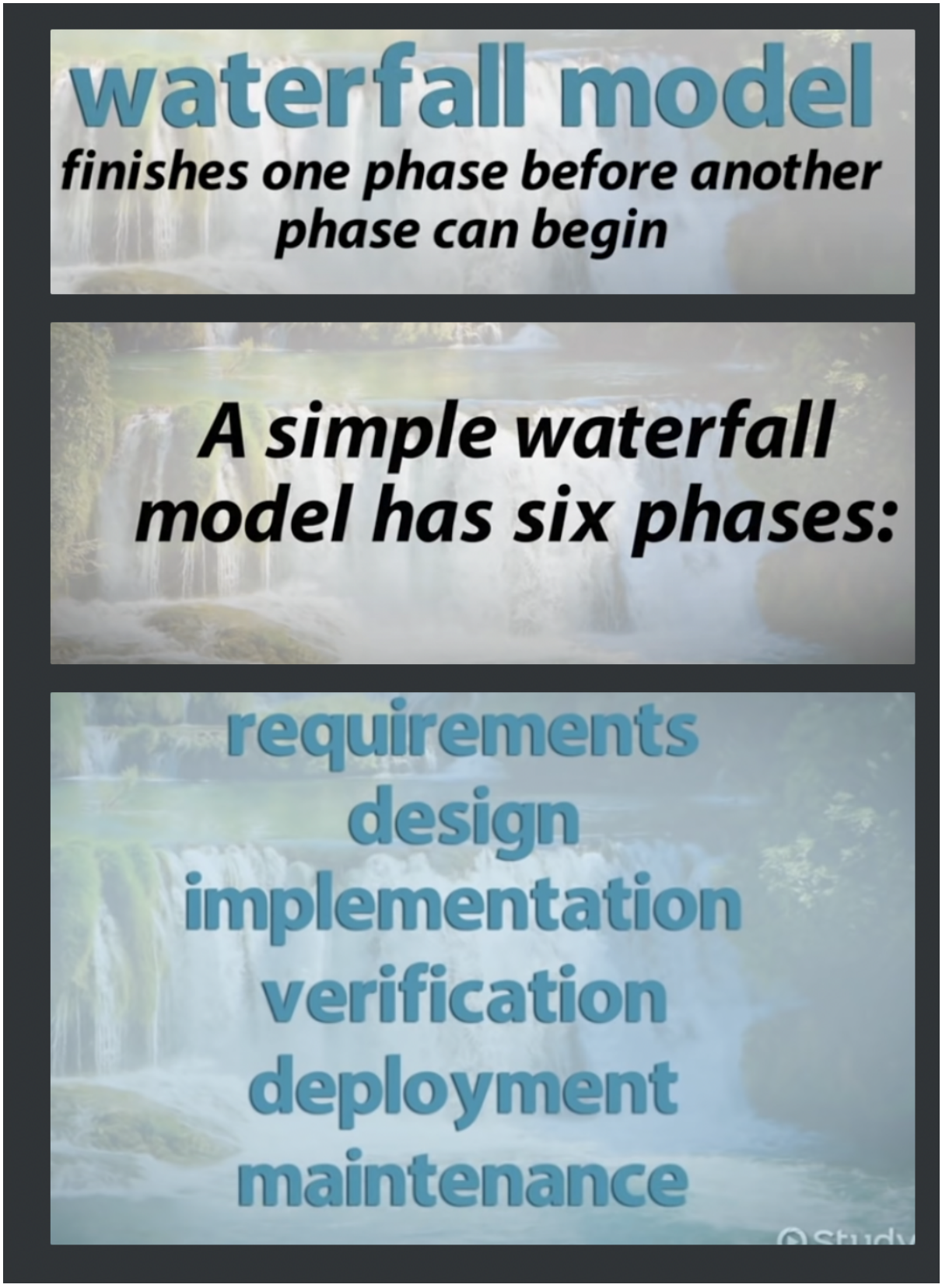


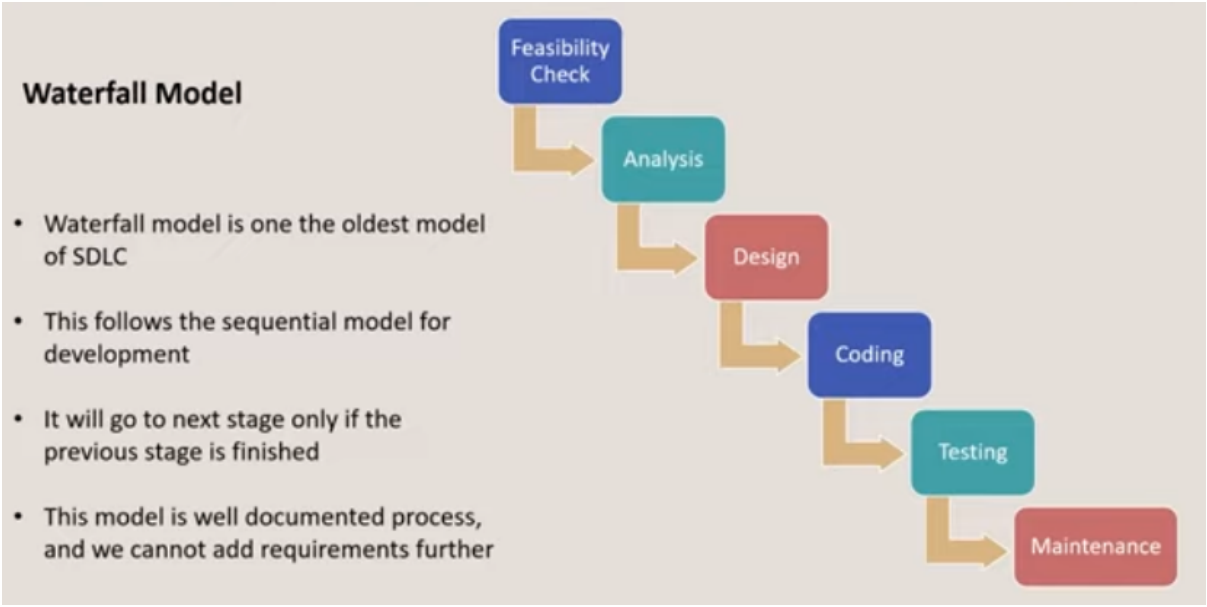

Phase 1
Formation
Phase 2
Requirement/
Planning
Phase 3
Design
Phase 4
Construct
Phase 5
Test
Phase 6
Product
Release
Phase 7
Post
Implementation
Project Initiation
Requirements Definition
Design
Construct
Test
Product Release
Enhancement, Maintenance
Rough Order of Magnitude (ROM) Project Development Estimate
Project Management Plan (Charter)
Internal/ External
Prototype
System Test
Operational Acceptance
Project Implementation Notice
SDLC Phases.png
Functional Requirements
Design Review
Code Review
Test Summary
Acceptance Document
Technical Requirements
Detailed Project Development
TRAINING
Requirements Review & Approval
DOCUMENTATION
Statement Of Work
Change of Scope Document
Change Control Process
Change Control Log

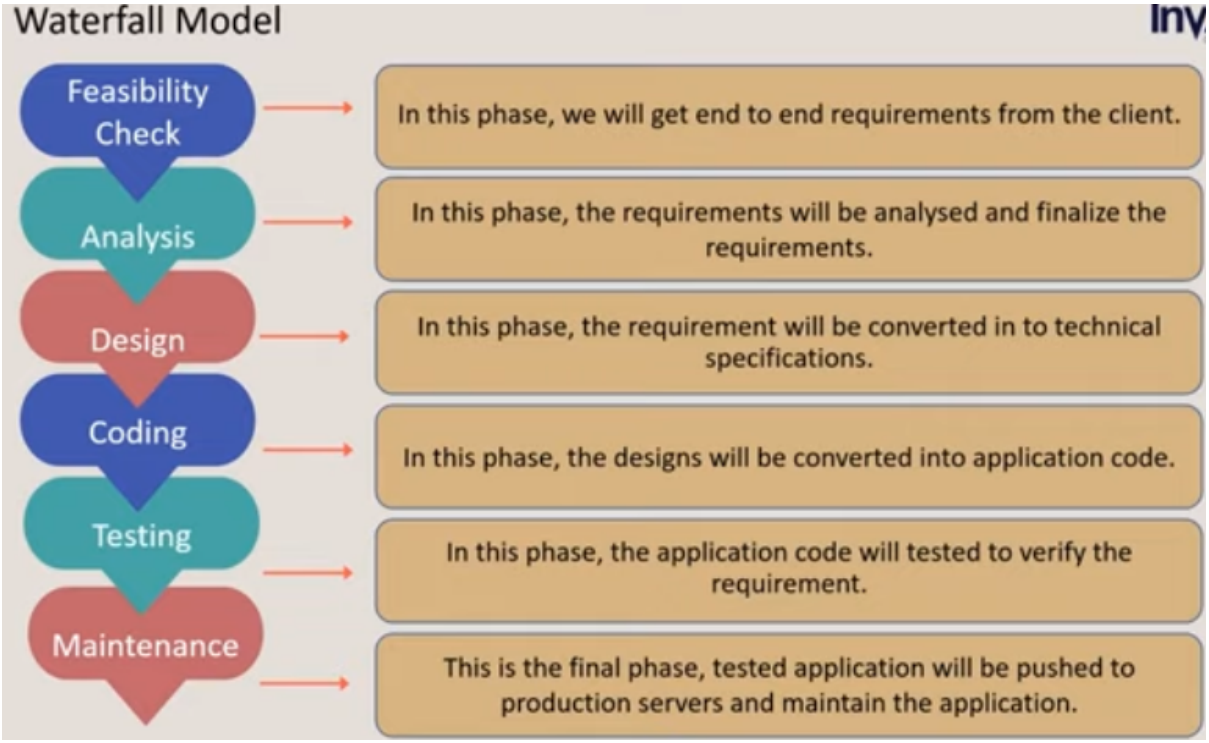

Open the window and make your first step in a new world!
Windows
Wolfi Korn


Types of Software Development Lifecycles
Agile Model
Agile Model is used for shorter releases based on priority.
DevOps
DevOps Methodology is used to maintain team coordination.
Waterfall Model
Waterfall Model is the oldest model of software development.
Lean Methodology
Lean Methodology works on principles to improve quality.


waterfall model
a sequential project management methodology where one phase completely finishes before the next phase begins


waterfall model
finishes one phase before another phase can begin
A simple waterfall model has six phases:
requirements
design
implementation
verification
deployment
maintenance


Waterfall Model
Waterfall model is one the oldest model of SDLC
This follows the sequential model for development
It will go to next stage only if the previous stage is finished
This model is well documented process, and we cannot add requirements further
Feasibility Check
Analysis
Design
Coding
Testing
Maintenance

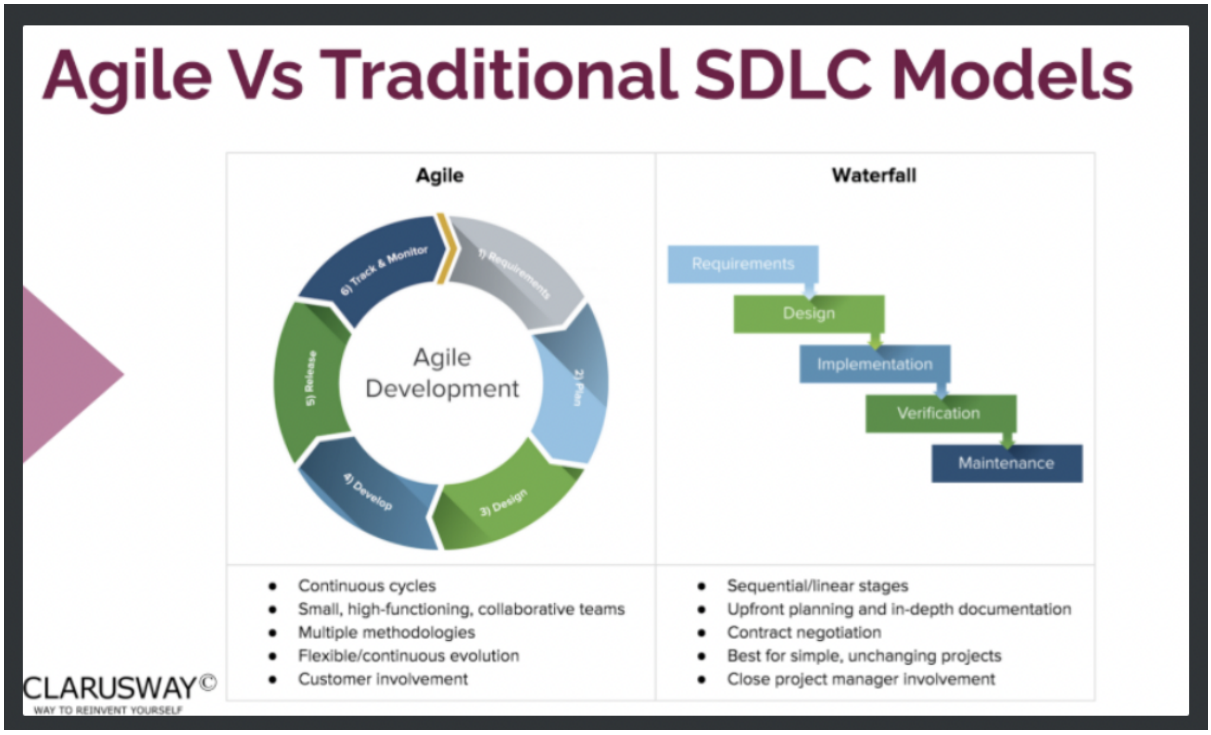

Waterfall Model
Feasibility Check
In this phase, we will get end to end requirements from the client.
Analysis
In this phase, the requirements will be analysed and finalize the requirements.
Design
In this phase, the requirement will be converted in to technical specifications.
Coding
In this phase, the designs will be converted into application code.
Testing
In this phase, the application code will tested to verify the requirement.
Maintenance
This is the final phase, tested application will be pushed to production servers and maintain the application.

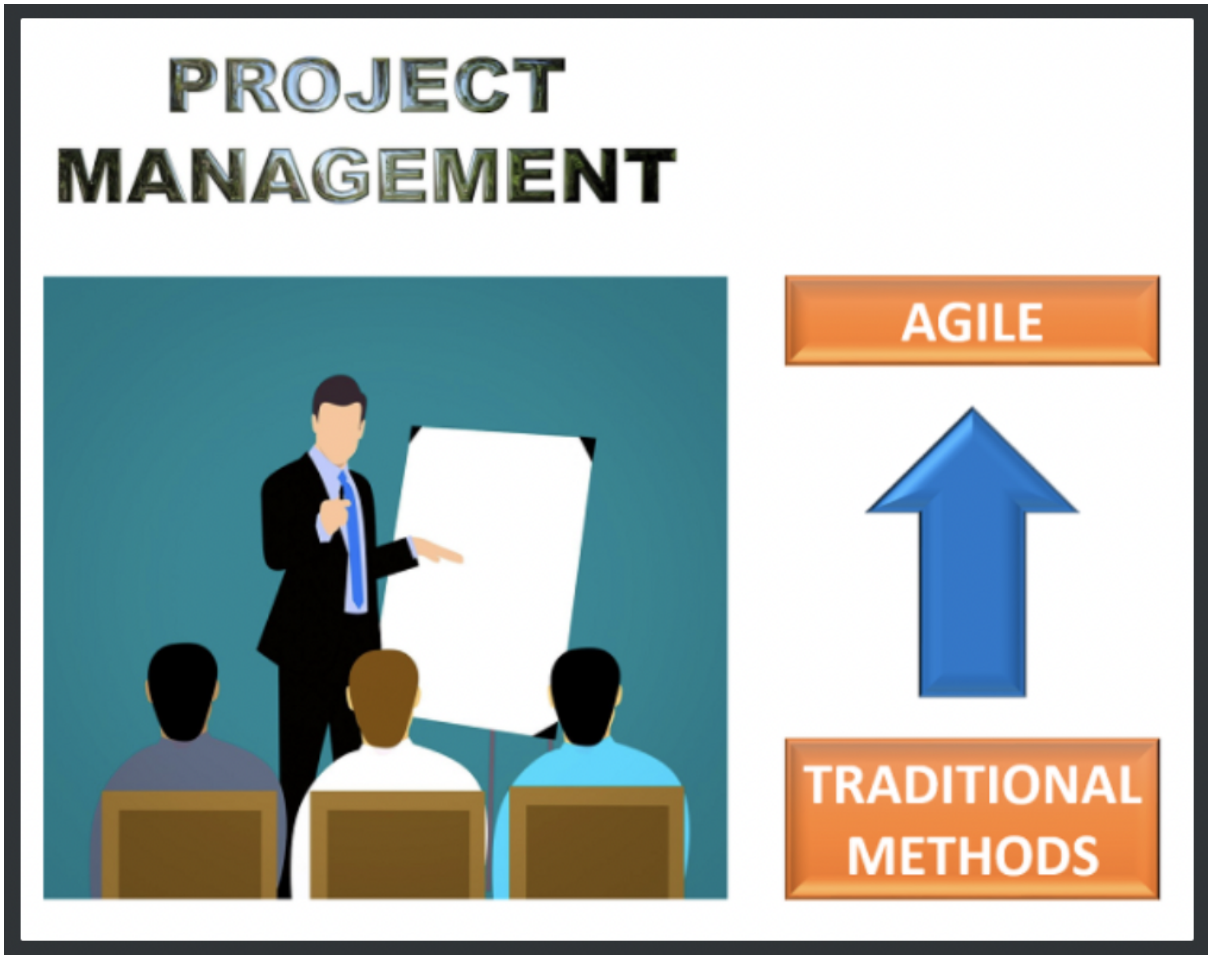
Advantages:

- easy to understand
- easy to manage
- fewer production issues
- better budget management

Disadvantages:

- not flexible
- doesn't handle unexpected risks well
- not a good for complex or long-term projects
- difficult to capture all requirements up front





The Agile Manifesto

In 2001, 17 independent Software Leaders met in America for brainstorming. They aimed to find out how to develop software better by using different knowledge and approaches. After two days of brainstorming, they released the Agile Manifesto.

The Agile Manifesto was a powerful statement, carefully crafted using only 68 words. Everyone agreed that the Agile Manifesto was both short and authoritative. While traditional methods advocated a stable plan and avoided changes, the manifesto focused on people, communication, the product, and flexibility.

The Agile Manifesto

A group of 17 people thought:

“We’re all doing these different approaches to developing software. We ought to get together and see where there are commonalities in what we’re thinking about.”

The result was a meeting at a ski resort in Snowbird, Utah in 2001.

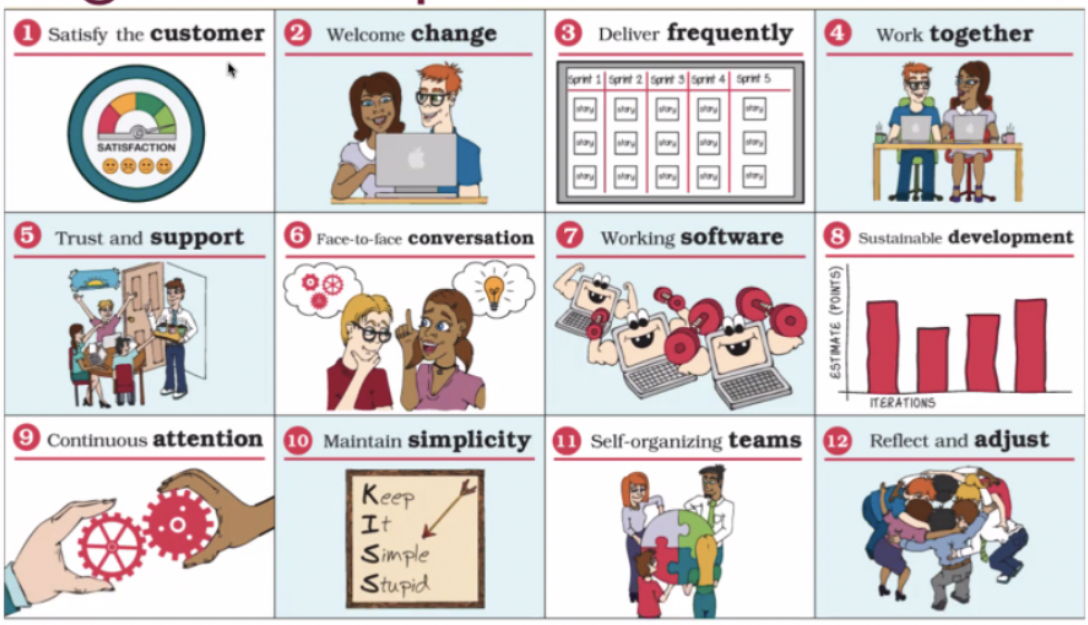

Manifesto for Agile Software Development
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
That is, while there is value in the items on the right, we value the items on the left more.
Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Agile Principles

In the months following the publication of the Agile Manifesto, the original signatories continued to communicate. They augmented the four values of the manifesto with the following 12 principles.

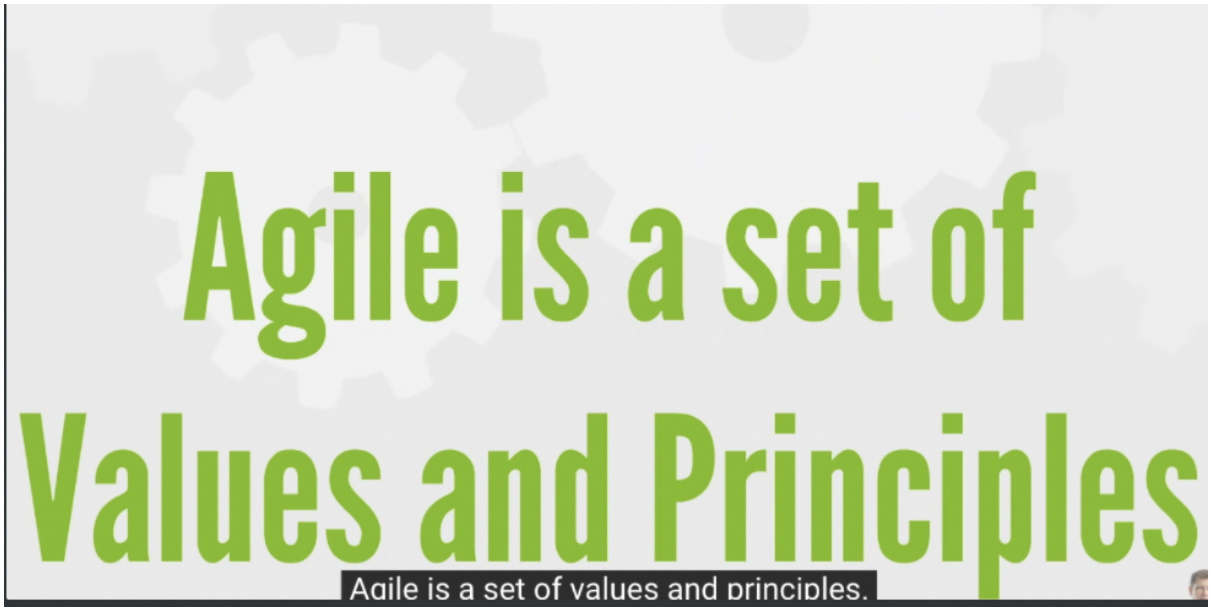
1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with preference to the **shorter timescale**.
4. Business people and developers must **work together** daily throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. 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**.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a **constant pace** indefinitely.
9. **Continuous attention** to technical excellence and **good design** enhances agility.
10. Simplicity -the art of **maximizing the amount of work** not done- is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At **regular intervals**, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

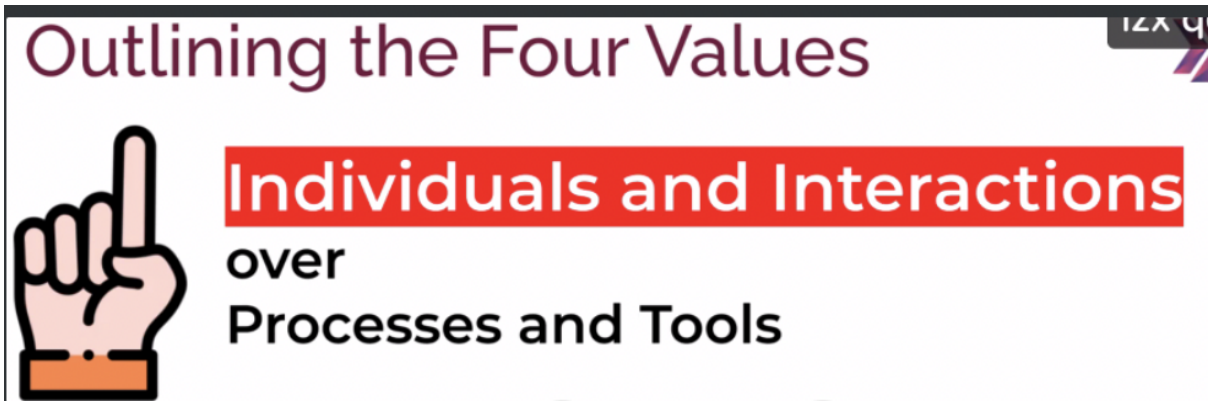
12 Agile Principles



Q: Do you know about Agile Manifesto & its Principles? Explain in brief.

A: There are four values in the manifesto. Individuals and interactions, working software, customer collaboration and responding to the changes are the values. Stemming from these values there are 12 principles in agile. These principles can be summarized as to satisfy the customer, to welcome changing requirements, good cooperation between business people and developers (working together), face to face conversation, motivated individuals and simplicity.

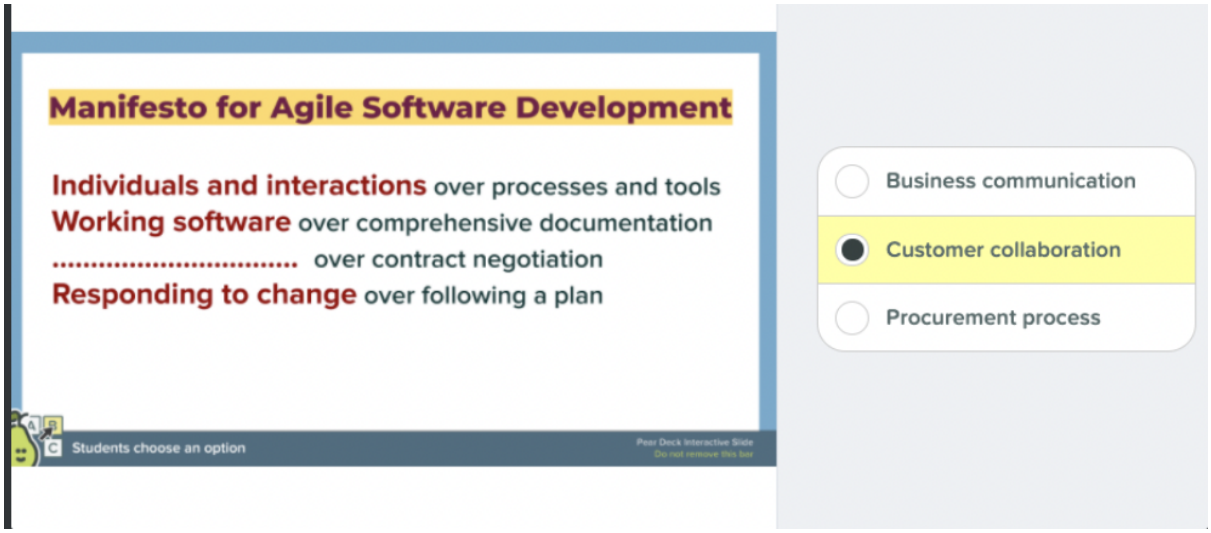

Agile is a set of
Values and Principles
Agile is a set of values and principles.

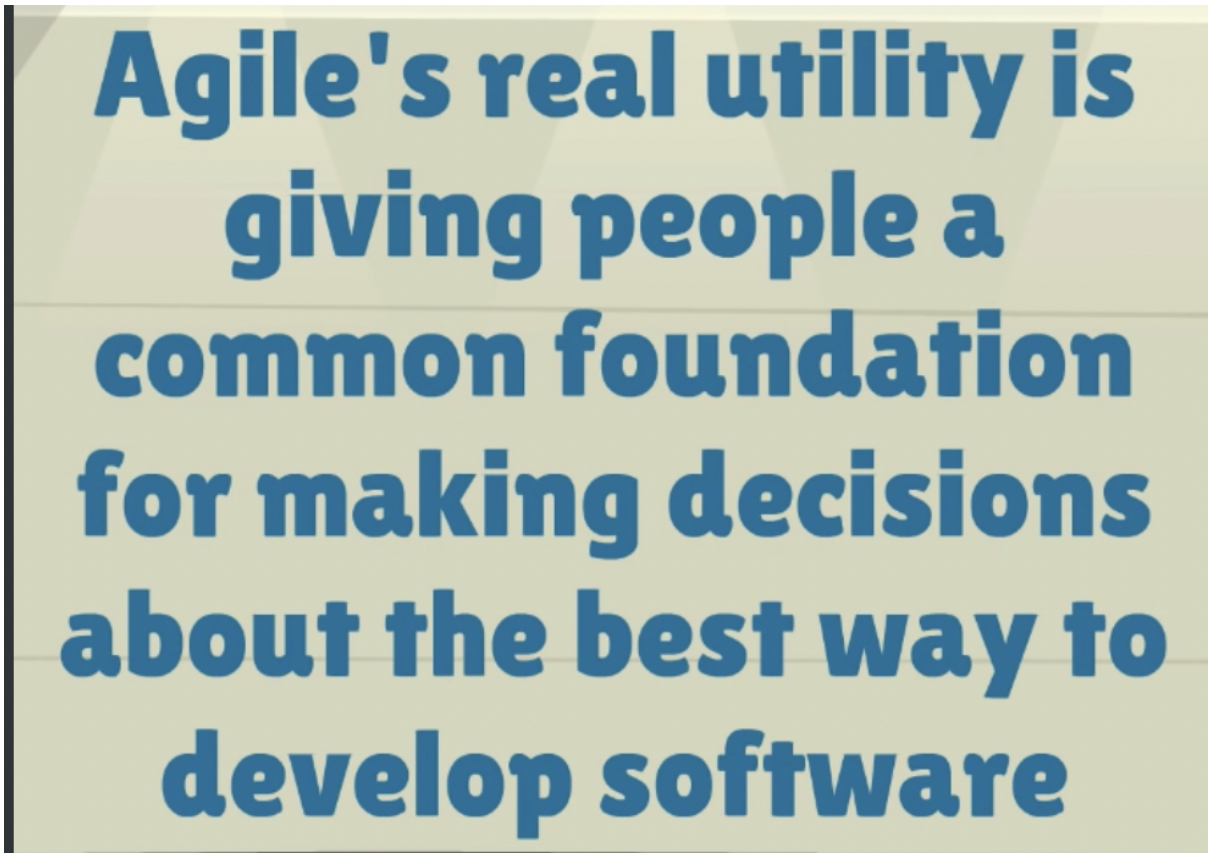

Outlining the Four Values
Individuals and Interactions
over
Processes and Tools

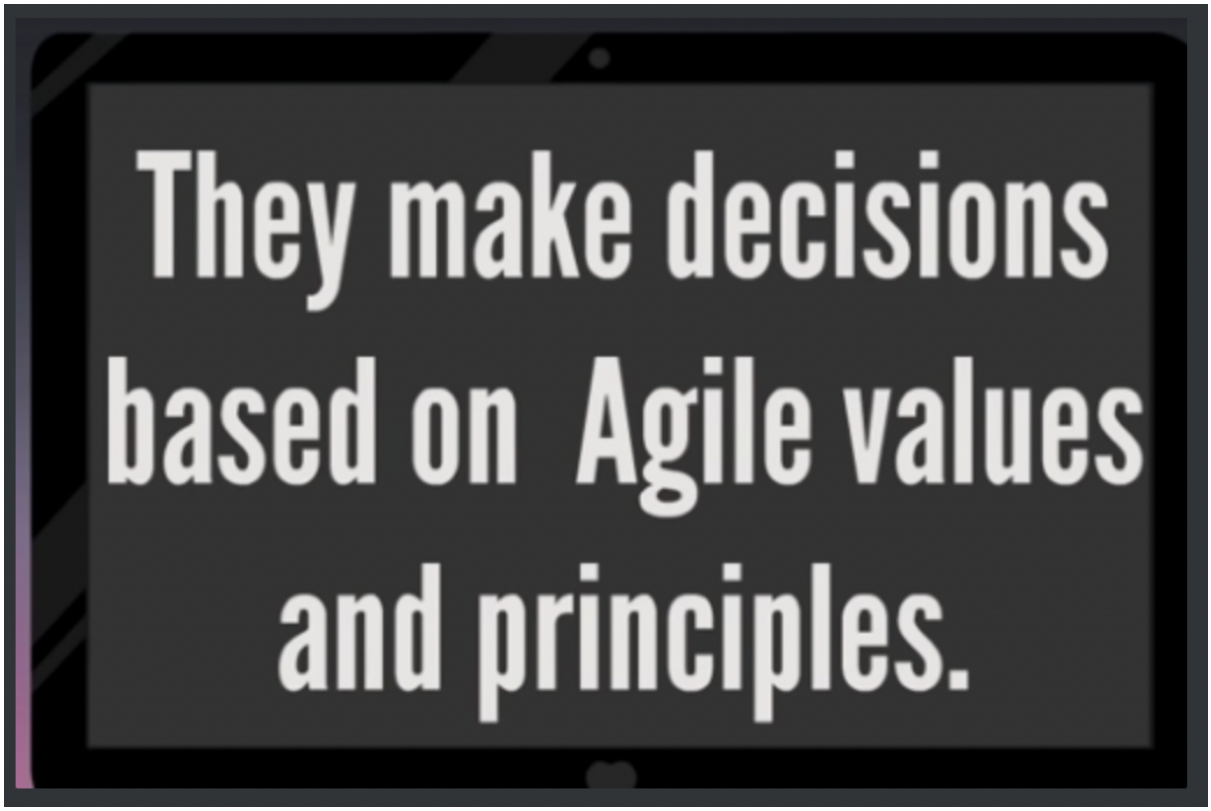

Working Software
over
Comprehensive Documentation


Customer collaboration
over
Contract Negotiation


Responding to change
over
Following a Plan

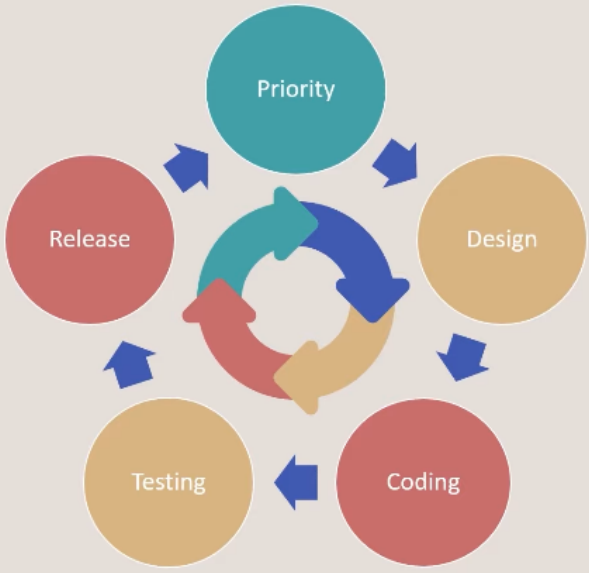

Manifesto for Agile Software Development
Individuals and interactions over processes and tools
Working software over comprehensive documentation
..... over contract negotiation
Responding to change over following a plan
Students choose an option
Business communication
Customer collaboration
Procurement process

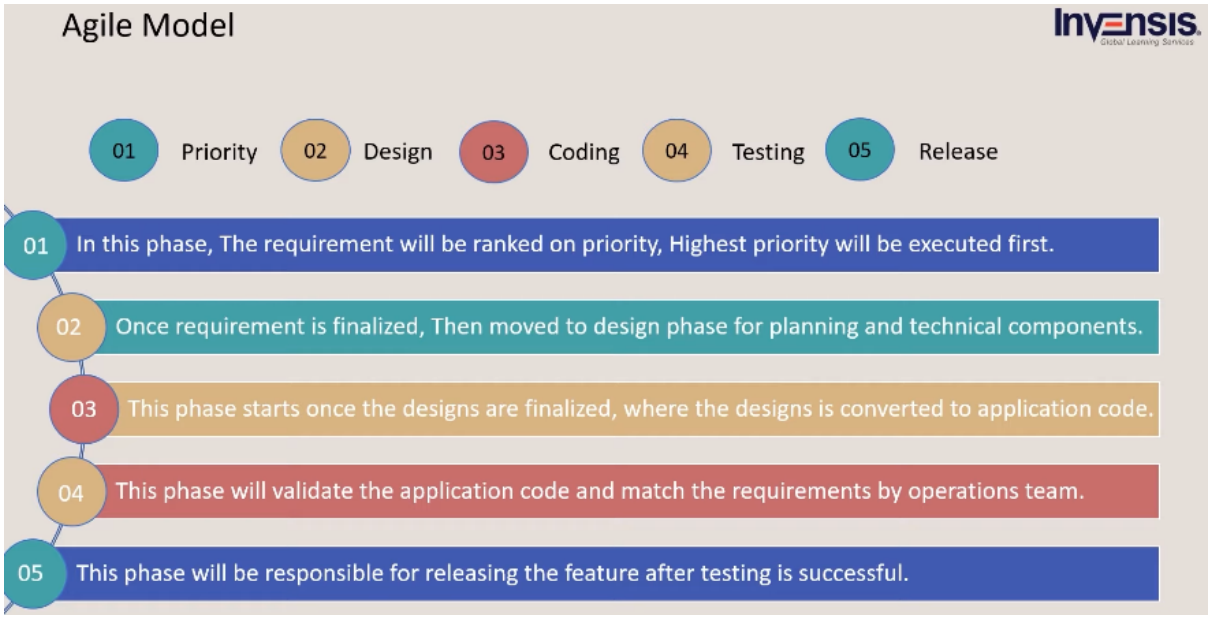

Agile's real utility is giving people a common foundation for making decisions about the best way to develop software

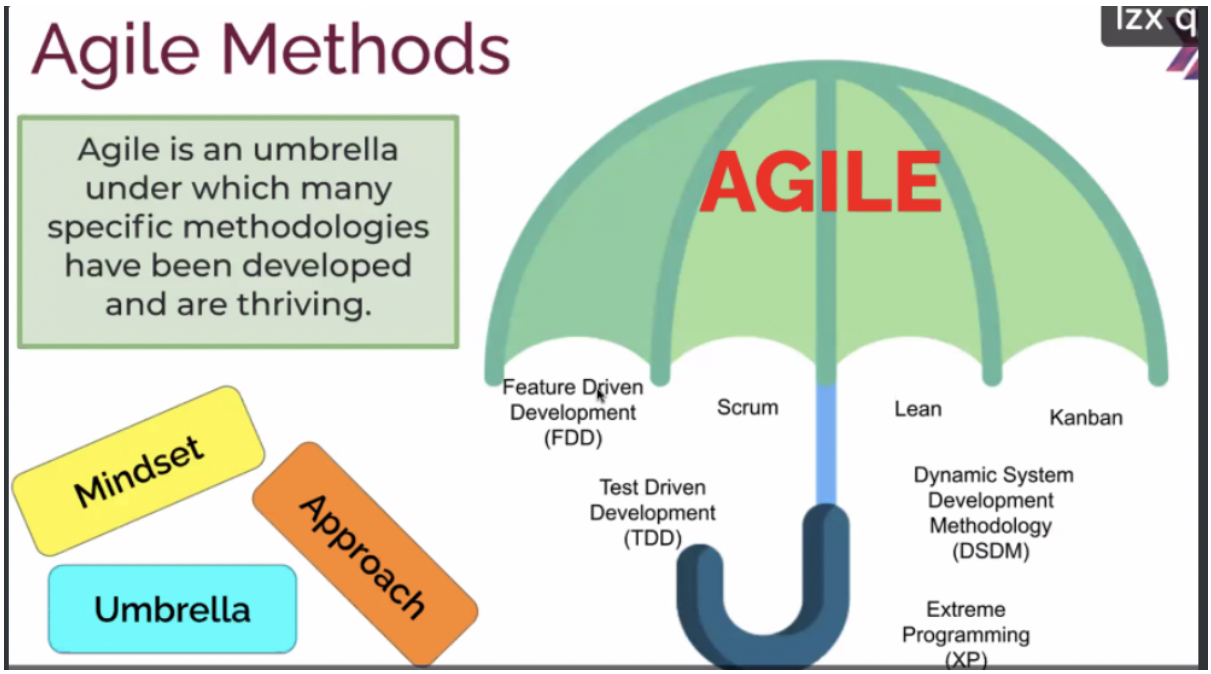


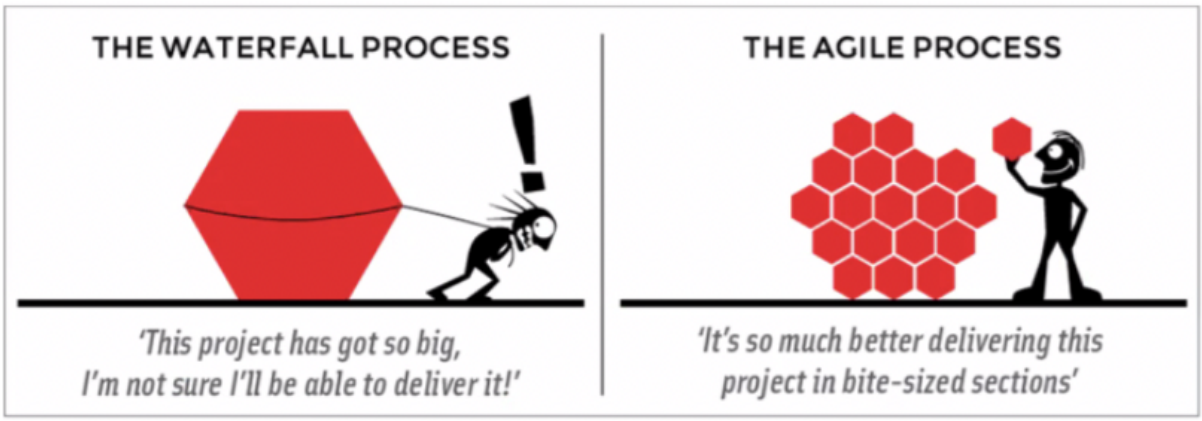
Agile Model

- Agile model is used to overcome the problems of waterfall model
- Agile is used for shorter releases
- This model will work on the priority based on client requirement
- This model is used to deliver the product in faster releases




Agile Model
Invensis
01 Priority
02 Design
03 Coding
04 Testing
05 Release
01 In this phase, The requirement will be ranked on priority, Highest priority will be executed first.
02 Once requirement is finalized, Then moved to design phase for planning and technical components.
03 This phase starts once the designs are finalized, where the designs is converted to application code.
04 This phase will validate the application code and match the requirements by operations team.
05 This phase will be responsible for releasing the feature after testing is successful.


Agile Methods
Agile is an umbrella under which many specific methodologies have been developed and are thriving.
AGILE
Feature Driven Development (FDD)
Scrum
Lean
Kanban
Test Driven Development (TDD)
Dynamic System Development Methodology (DSDM)
Extreme Programming (XP)
Mindset
Approach
Umbrella


THE WATERFALL PROCESS
'This project has got so big, I'm not sure I'll be able to deliver it!'
THE AGILE PROCESS
'It's so much better delivering this project in bite-sized sections'

Agile Manifesto

Agile Values

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

Individuals and interactions over **processes and tools**

Working software over **comprehensive documentation**

Customer collaboration over **contract negotiation**

Responding to change over **following a plan**


SCRUM

What is Scrum?

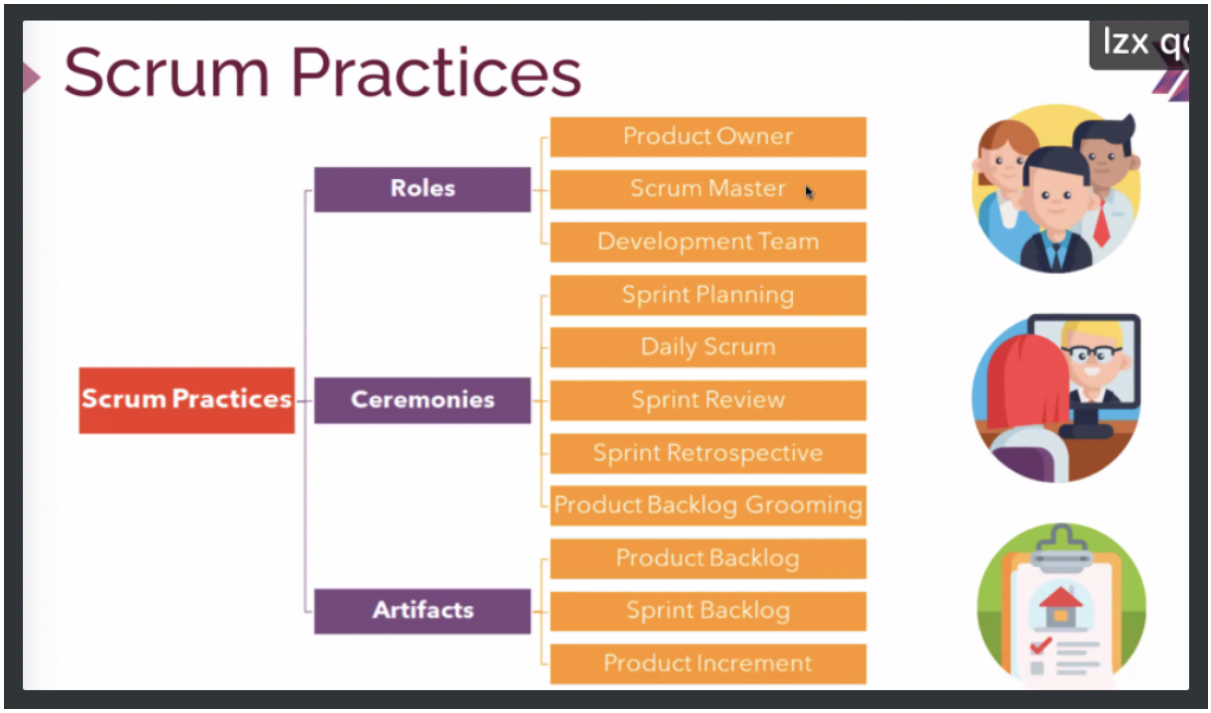
A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

Lightweight | Simple to understand | Difficult to master

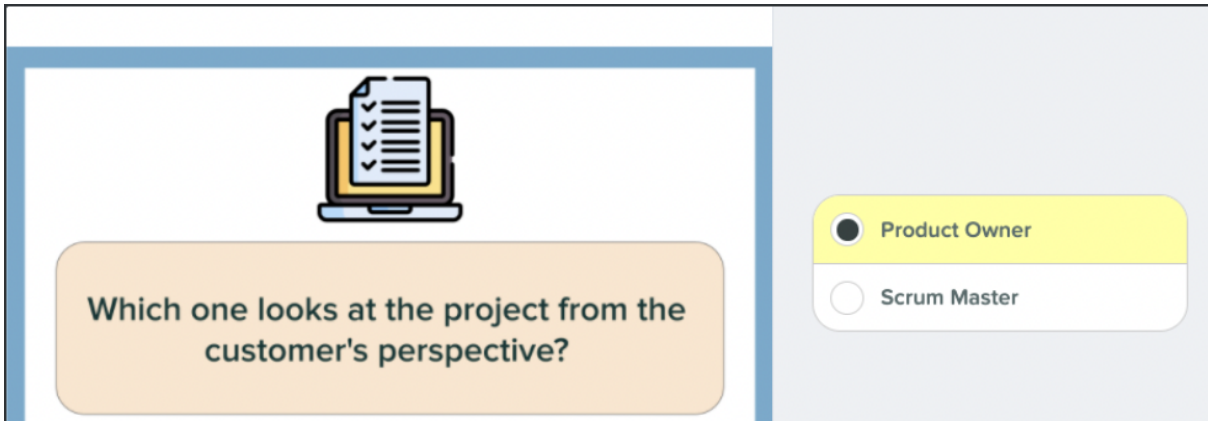
What is Scrum?

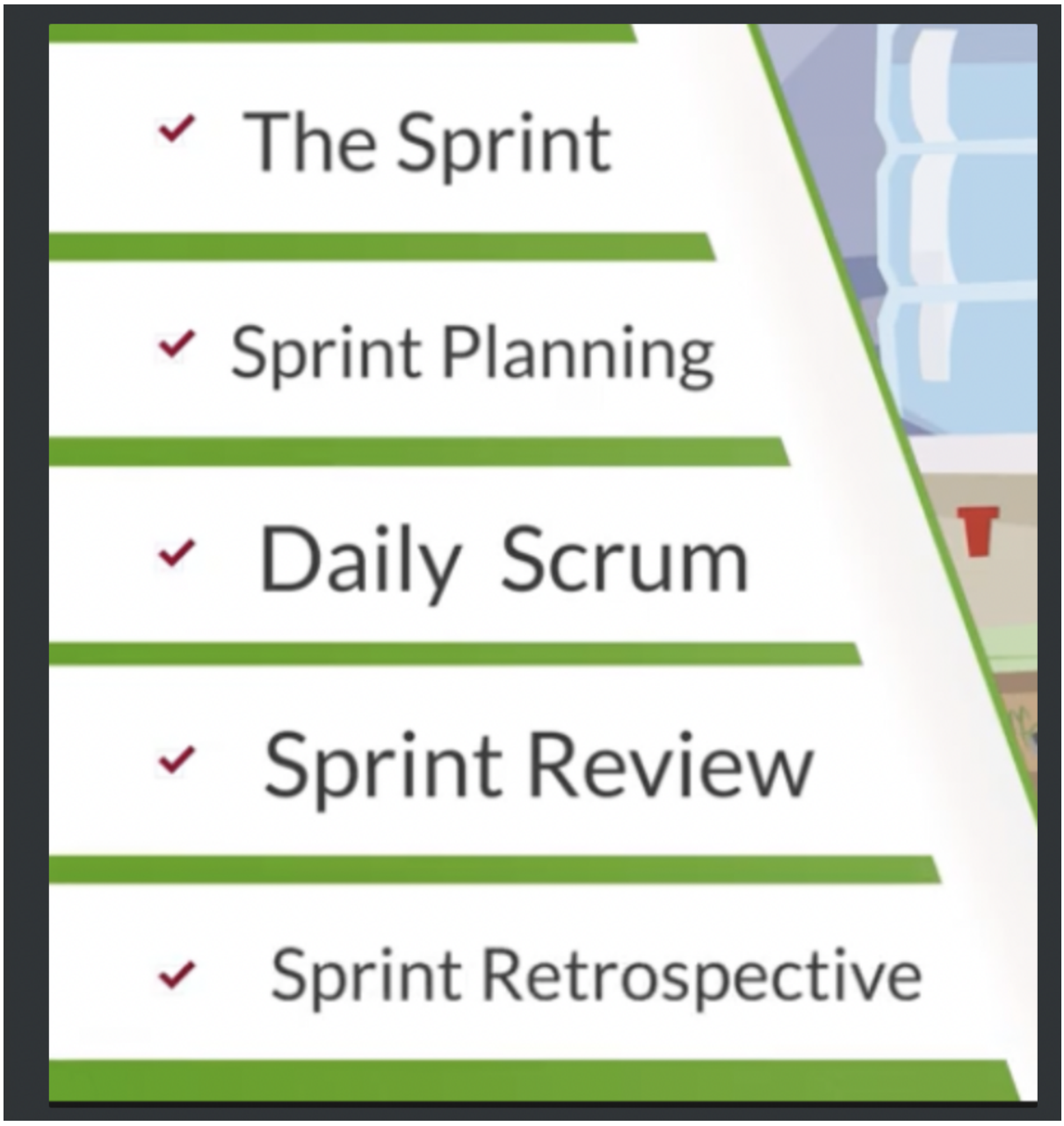
Scrum emphasizes delivering business value frequently through short iterations known as sprints.

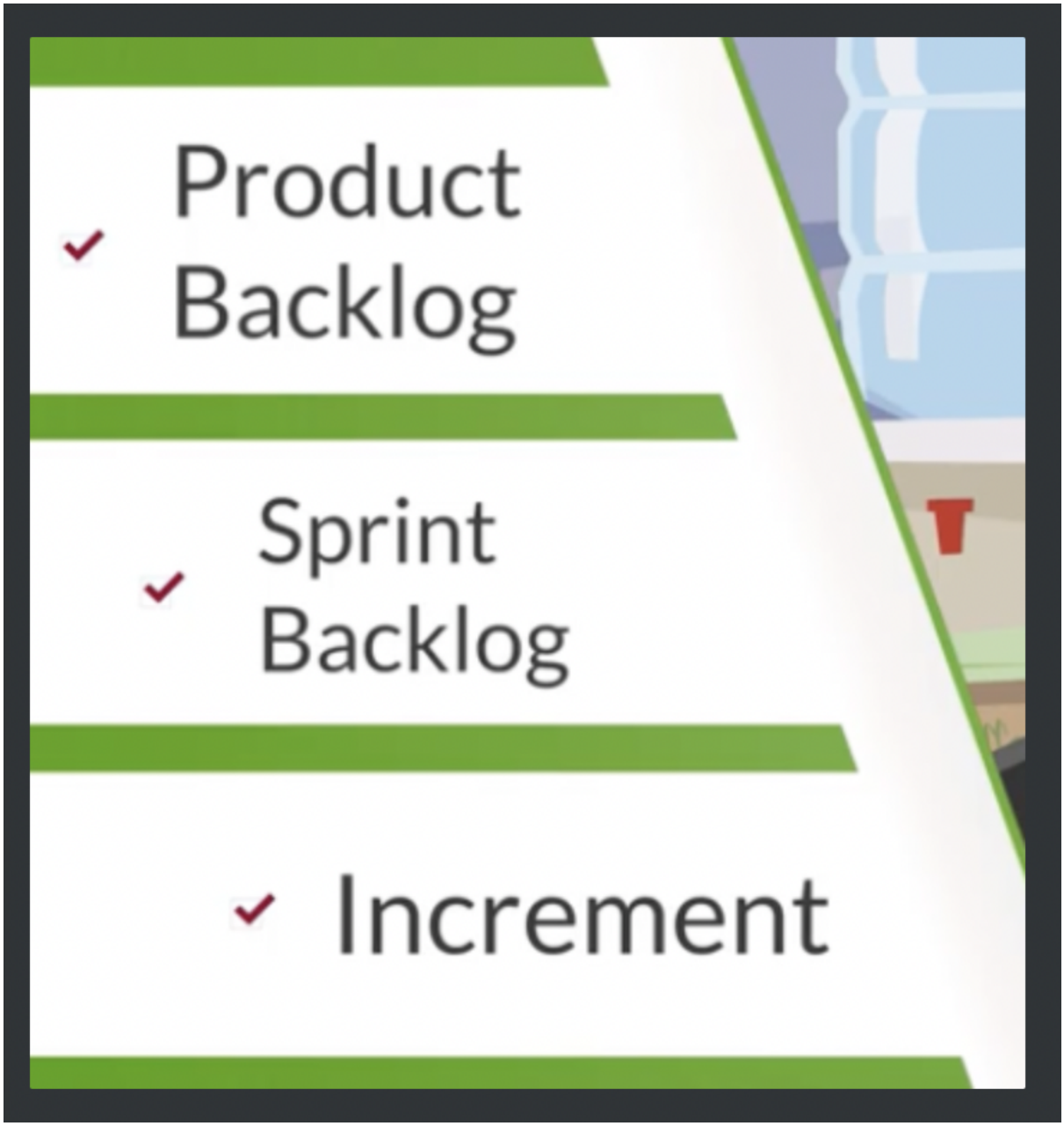
This gives visibility to the work that's being done and creates opportunities for feedback.


Scrum Practices
Scrum Practices
Roles
Product Owner
Scrum Master
Development Team
Ceremonies
Sprint Planning
Daily Scrum
Sprint Review
Sprint Retrospective
Product Backlog Grooming
Artifacts
Product Backlog
Sprint Backlog
Product Increment


Scrum Roles
Product Owner
Responsible for the project's success by defining the project vision, requirements, and priorities
Scrum Master
Accountable to the team to remove impediments that will prevent them from achieving the goals of the Product Owner
Development Team
Team comprises 3-9 people, with a mix of roles, and self-organizes to determine how to best meet the goals of the Product Owner


Which one looks at the project from the customer's perspective?
Product Owner
Scrum Master


The Sprint
Sprint Planning
Daily Scrum
Sprint Review
Sprint Retrospective


Product Backlog
Sprint Backlog
Increment

Scrum Framework and Sprint Concept

Scrum is the most common methodology to implement the Agile. It is an iterative development model used in complex software development processes. In scrum, larger projects are divided into smaller parts that can be managed with **sprints** . Sprints are the periods from one to four weeks. It can be even a few days when needed. Steady sprint length reduces variability; a scrum team can safely predict what they can do on each sprint based on what they have done in previous sprints. The implementation of sprints allows scrum teams to make arrangements for instant improvement, rather than at the end of the project. At the end of each sprint, something remarkable is revealed. For gaining feedback from users or investors, the product produced during each iteration should be demonstrated. The scrum framework defines specific roles, artifacts, and activities for projects. The following figure shows all of these components of the scrum framework that we will discuss later one by one. |

> **Q**: What is the duration of a scrum sprint?
> **A**: It depends on the number of people in the development team and the size of the project. In general, a scrum sprint is completed in 1-4 weeks.

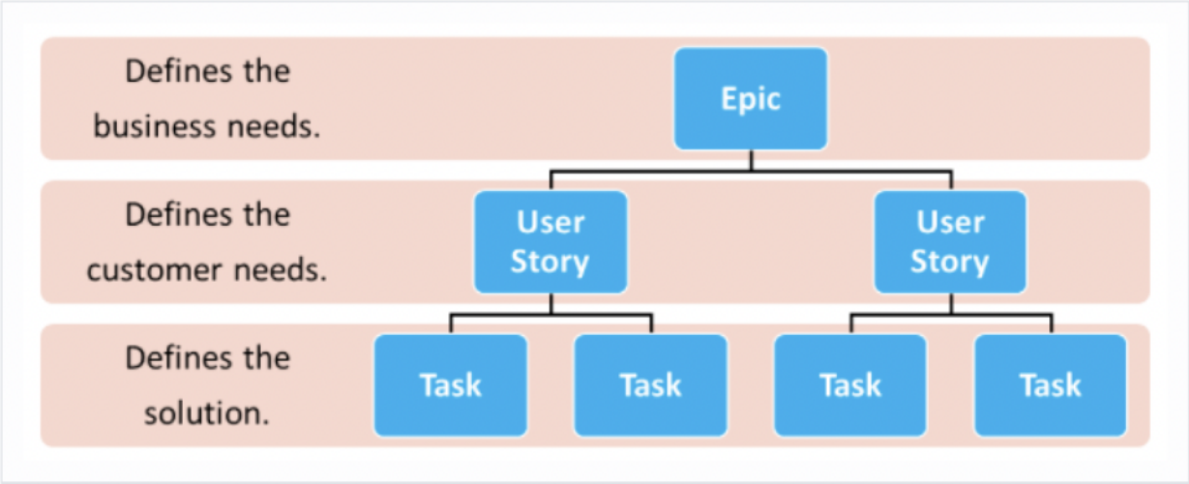
Scrum Roles

There are three main roles in scrum projects. These are the **Product Owner**, **Scrum Master**, and **Development Team**.

Product Owner (PO) is the business representative in the team and speaks for the needs of the project for maximizing the value delivered in each sprint. The product owner represents stakeholders and is the voice of the customer. Therefore, the product owner works together with stakeholders and prioritizes the product requirements.

Scrum Master coaches the team, protects the team from organizational distraction, clears any obstacles encountered and helps team members focus on what they do. Scrum Master ensures that scrum is understood well by the team members and it is working properly. Scrum Master constantly improves the team's environment. While the product owner has a directing role, Scrum Master has an enabling role in a scrum team.

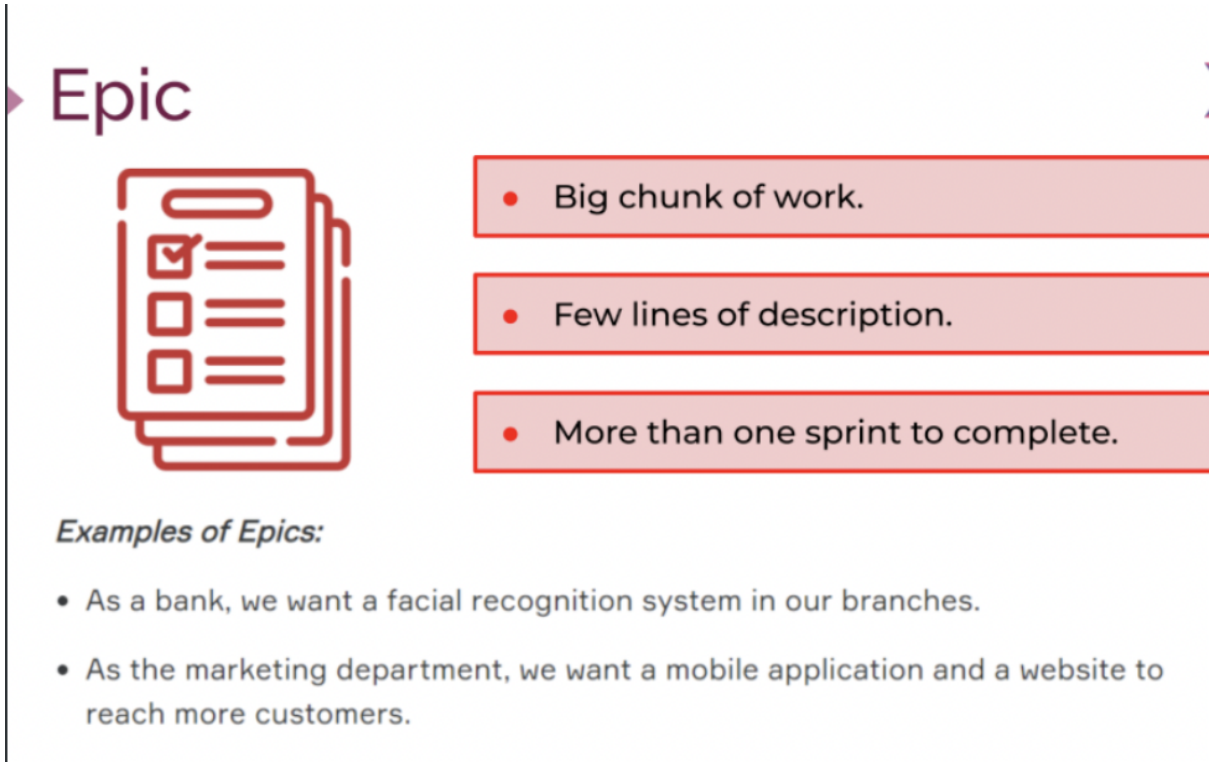
A **Development team** usually consists of 3-9 people and performs daily tasks. The team is project-oriented and dedicated to the success of the project. Each team member is very talented that is, the team members are skilled in certain subjects. Each member can do more than one job on the project. Discipline and integrity are the key terms for a successful team.



Hierarchy Between Epic, User Story and Task

Examples of Epics:

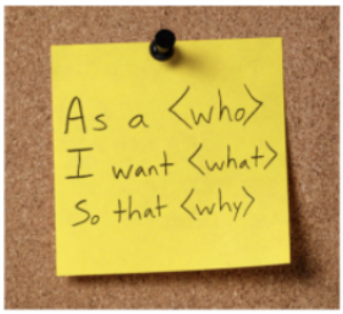
- As a bank, we want a facial recognition system in our branches.
- As the marketing department, we want a mobile application and a website to reach more customers.



💡Tips:

- Template of a User Story:
  As a < type of user >, I want < some goal > so that < some reason>

User Story

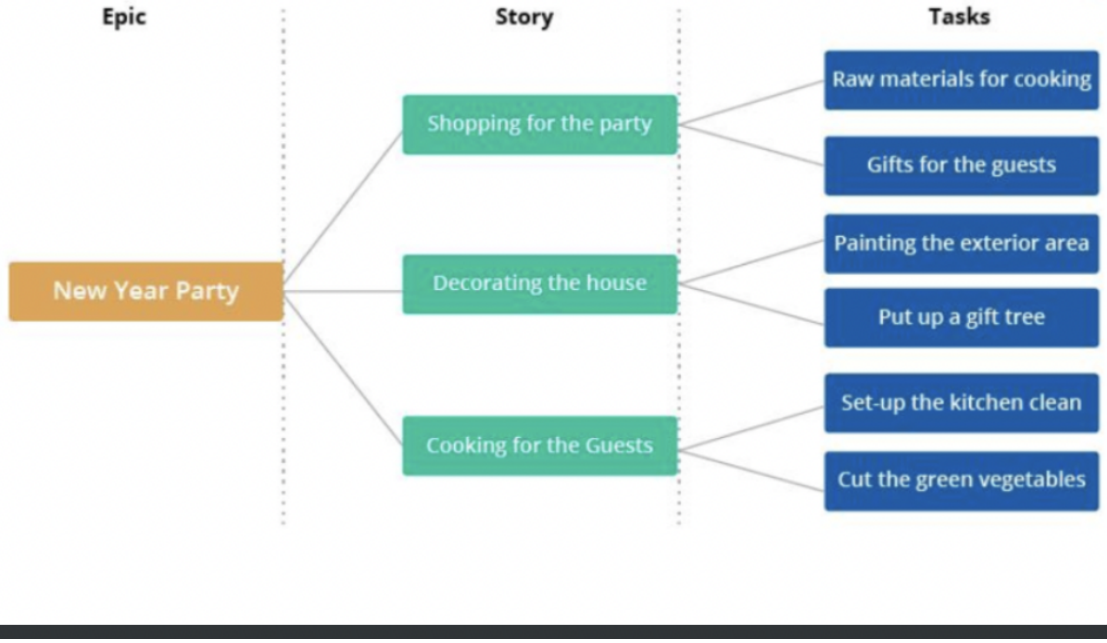


- User Needs.
- Few lines of description.
- Deliver during a sprint.

Examples of User Stories:

- As a registered user, I want to add items to the cart so that I can purchase multiple items at once.
- As a student, I want to apply for the exam online so that I can save time.

Epic, User Story and Task

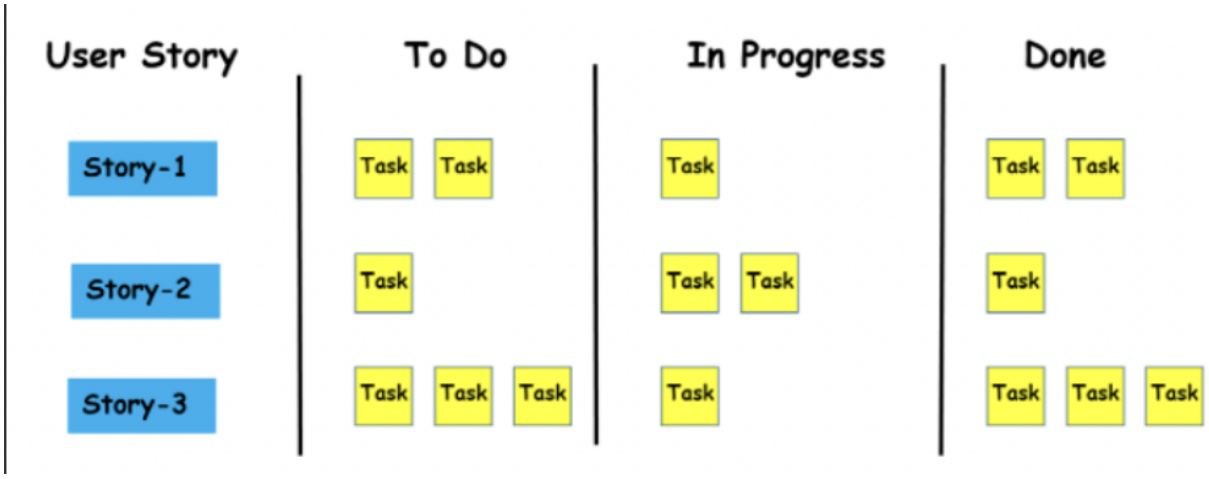


Task

- Represents a technical activity
- Description of individual work item
- Created by anyone

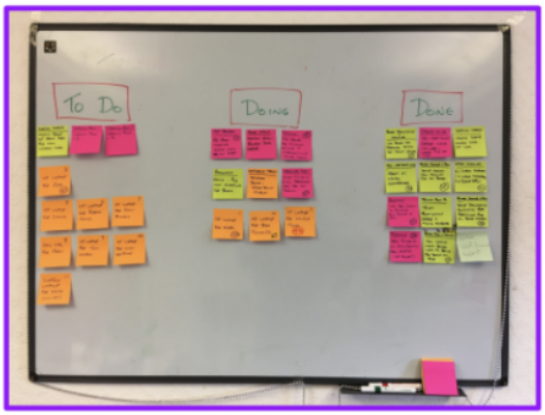
Examples of Tasks:

- Redesign a single web page
- Create a new logo
- Perform usability testing

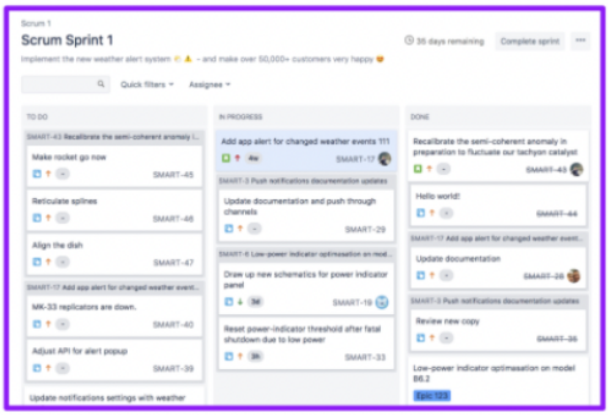


Scrum Board

Physical Scrum Board



Online Scrum Board



Product Backlog

The product backlog refers to the **list of everything that needs to be done to complete the project**. Beside all user stories, it also includes technical tasks. The product backlog is the responsibility of the product owner. The product owner fulfills this responsibility by creating the product backlog, **prioritizing** the requirements in the product backlog list and constantly updating this list. The product owner updates the product backlog because once a story is completed, it should be removed from the list. Sometimes, however, new stories are added, as the project grows.

Sprint Backlog

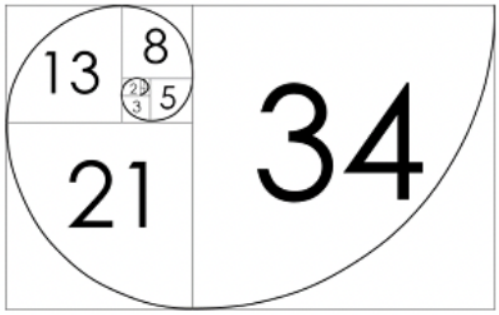
The sprint backlog can be defined as a subset of the product backlog. The sprint backlog is generated from the product backlog during the sprint planning meeting at the beginning of each sprint. The user stories selected from the product backlog, which will be completed during the sprint constitute the sprint backlog.

The sprint backlog is **not a flexible list** like a product backlog. That means the sprint backlog is unchanged during the sprint period. Once agreed upon in the sprint planning meeting, the stories, and steps to complete them remain stable during the sprint length. If there are stories left still unfinished by the end of the sprint, they will be added back to the product backlog and addressed during the next sprint.

Story Point

Story Points are decided upon and used by individual scrum teams. A Stor Point is a relative unit of measure to provide relative predictions of effort for completing tasks or user stories.

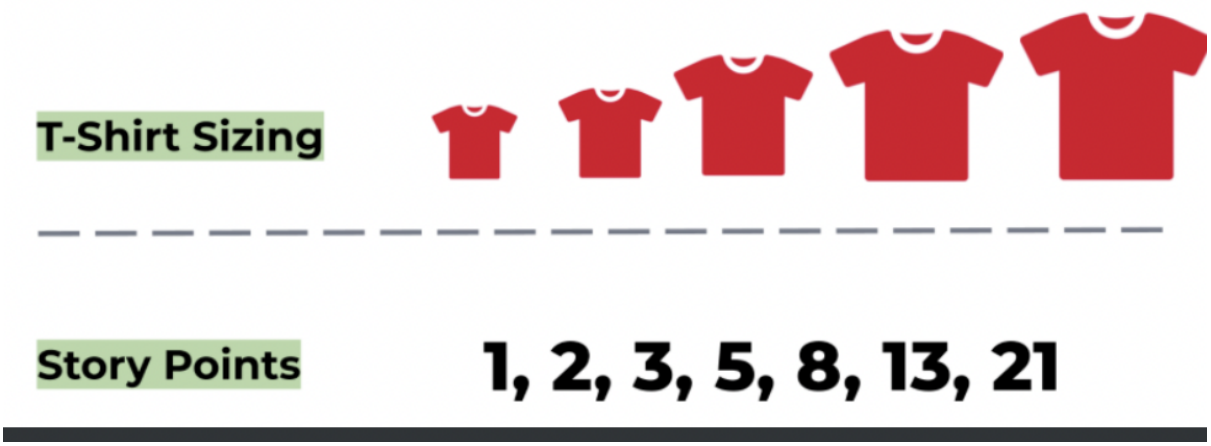
A Story Point provides an easier estimation to the team. Rather than assessing a product backlog item and estimating it in hours, teams consider only how much effort this item will require, relative to other product backlog items.



FibonacciNumbers

In other words, a story point is a numeric value that indicates the difficulty level of the user story. Before the development team makes an estimation, story points are assigned to each user story using the **Fibonacci numbers** (1, 2, 3, 5, 8, 13, 21, 34...). A story that is assigned 2 story points should be twice as difficult as a story that is assigned 1 story point.

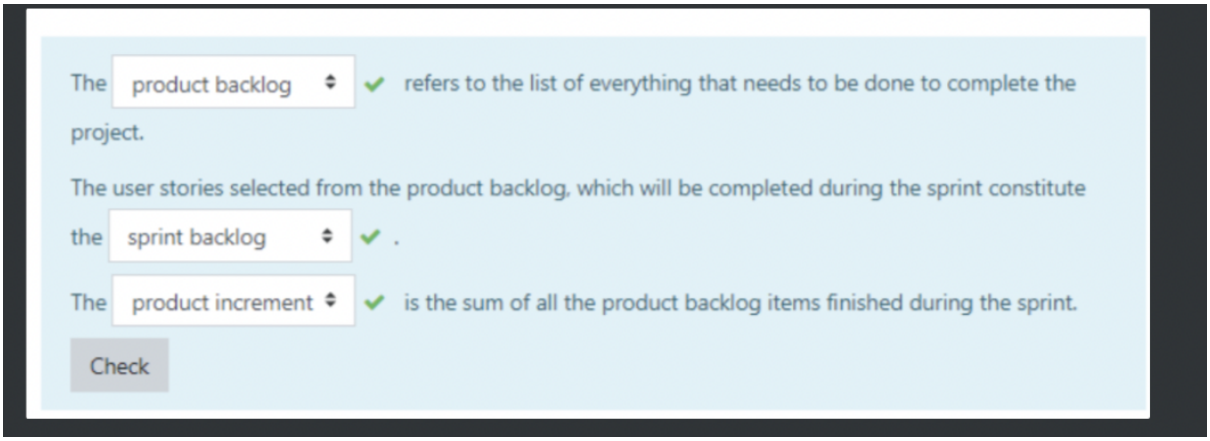
Estimation



Story Points Estimation Cheat Sheet

How much is known about the task	Everything	Almost everything	Something	Almost nothing	Nothing	Nothing
Dependencies	None	Almost none	Some	Few	More than few	Unknown
How much work effort	Less than 2 hours	Half a day	Up to two days	Few days	Around a week	More than one week
Story Points	**1**	**2**	**3**	**5**	**8** Should be split into smaller items	**13** Must be split into smaller items

> **Q**: Explain the term 'increment' in Scrum.
> **A**: The Product Increment is the sum of all the product backlog items finished during the sprint. In other words, by the end of each sprint, the development team creates a new software that gets built into the main product and this new software is called product increment. The product increment aims to invest in small amounts in the new features of the main product. This helps to shorten the time before receiving feedback. As the name implies, product increment continues to increase within the subsequent sprints. That means each product increment includes all the previous sprint increment values as it is cumulative.



The Sprint Planning Meeting ✔ takes place at the start of each sprint and all the scrum roles take part in this meeting.

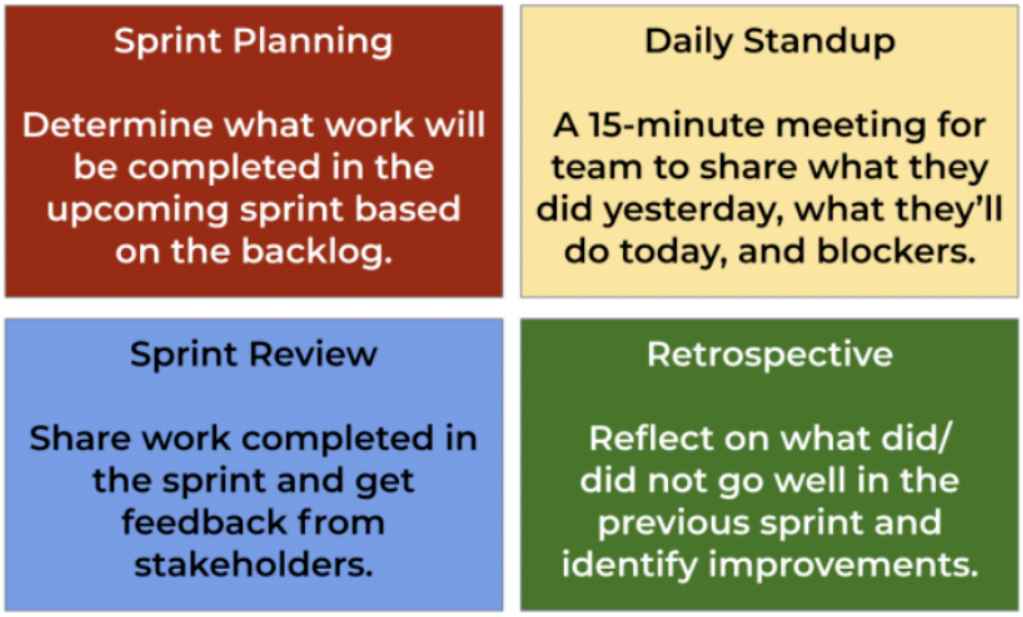
The Grooming Meeting ✔ is held to review the backlog

The maximum time allocated for the Daily Stand Up Meeting ✔ is 15 minutes.

In Sprint Review Meeting ✔ , the development team demonstrates to the whole organization what they accomplished during the sprint and receives feedback.

Like sprint review meetings, Sprint Retrospective Meeting ✔ is also held at the end of each sprint.

Scrum Meetings



Software Development Lifecycle Models (Continued)...

Lean Model

- Lean model is used to overcome the problems of Agile model
- Lean model concentrates on quality
- This model will work on continuous improvement
- This model works on 7 principles
- Drawback in this model is there is no team coordination

Software Development Lifecycle Models (Continued)...

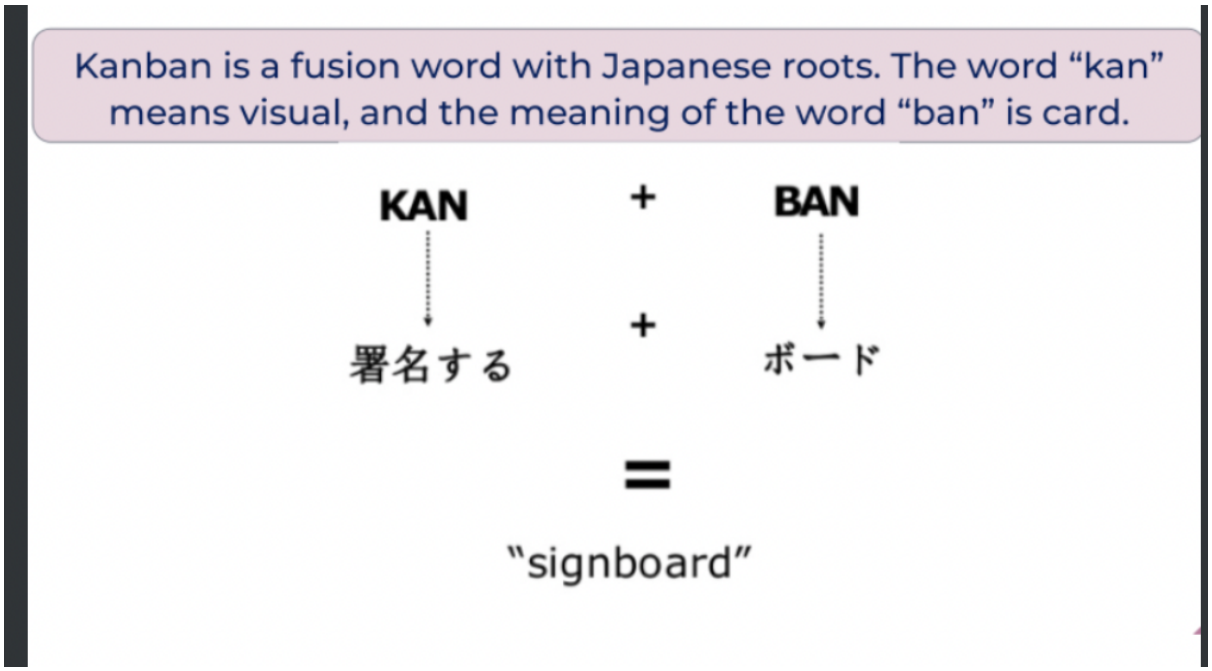
Waterfall Vs Agile Vs Lean Model

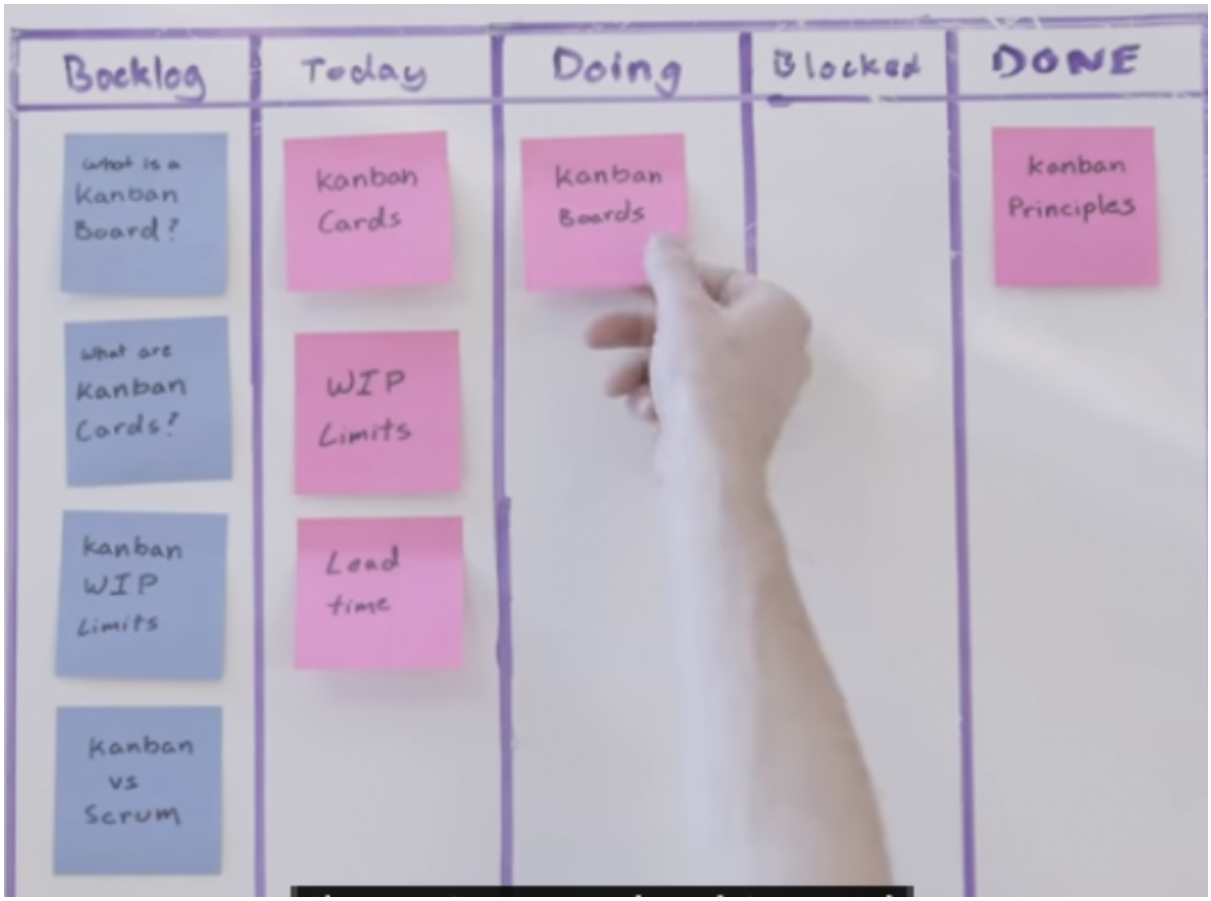
Features	Waterfall model	Agile methodology	Lean Model
Process	Follows Top-down approach or Sequential method	Involves processes like Agile Kanban, Agile Scrum	7 principles are involved to maintain constant improvement
Scope	Works well, if the project is short and simple.	For faster delivery of product, irrespective of quality.	Used for faster delivery with quality as major priority.
Feedback	Feedback is received only at major stages	Feedback is received at all points during development	Feedback is received at every stages during development but there can be delay in team coordination
Drawbacks	Depends on initial requirement, If requirement is wrong at the end by any manner will effect the project	As it does not have a definite plan and works on requirements, the product may not be the same as initially intended.	Everyone in the team should follow principles for quality of product, if someone does not follow then the impact on productivity is high.

Introduction to Kanban

Scrum is the most common way to implement the agile, however, kanban is another popular methodology for implementing the agile system in the business.

The word "kanban" is in Japanese and can be translated as "the card you can see" or "signboard". As the name implies, it is a visual framework used to implement agile and shows **what to produce**, **when to produce it**, and **how much to produce**.





> **Q**: Explain what is Kanban.
> **A**: A Kanban is like a flash card carrying all the information about the current status of your work and the work required to be done on the product at each stage of the software development process.

> **Q**: Describe the places where 'Scrum' and 'Kanban' are used?
> **A**: Scrum is a better choice when you need a more prominent process. However, if you want improvement in running the process without much changes in the whole scenario, you should use Kanban.

A Brief History of Kanban

Kanban's story dates back to the 1940s. During these years, Toyota updated its production method based on the model that supermarkets use to manage stocks on shelves. Supermarkets stock enough products to meet consumer demand. This is a method that optimizes the flow between the supermarket and the consumer. Since inventory levels match the consumption rate, the supermarket stores the optimum quantity of products at any given time.

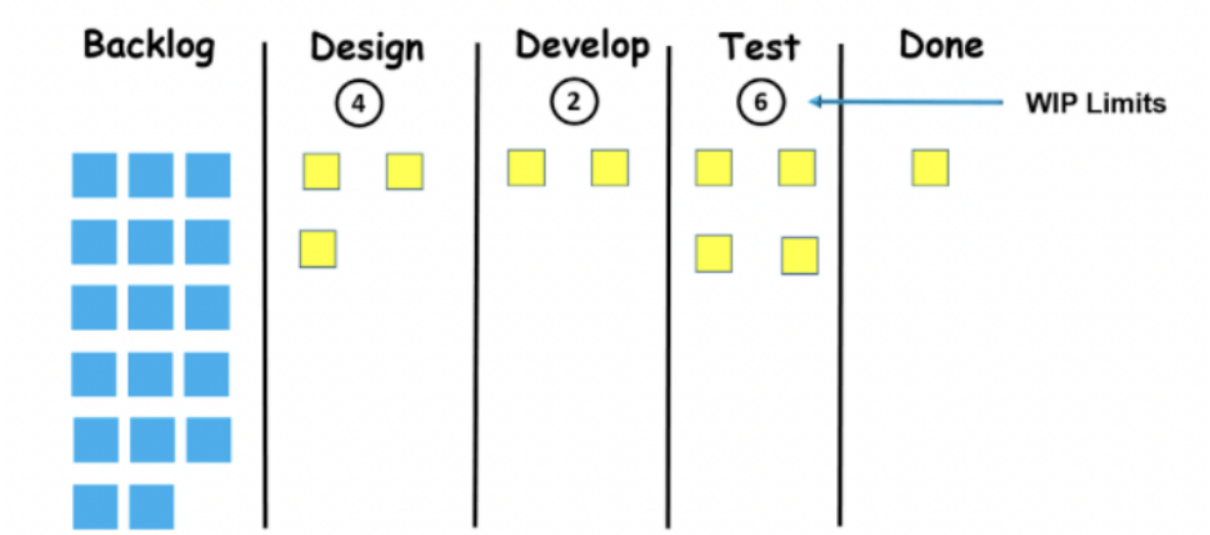
WIP Limits: The maximum number of cards that can be in a column at any given time is called WIP limits. WIP limits are written **on the top of each column** on the board.

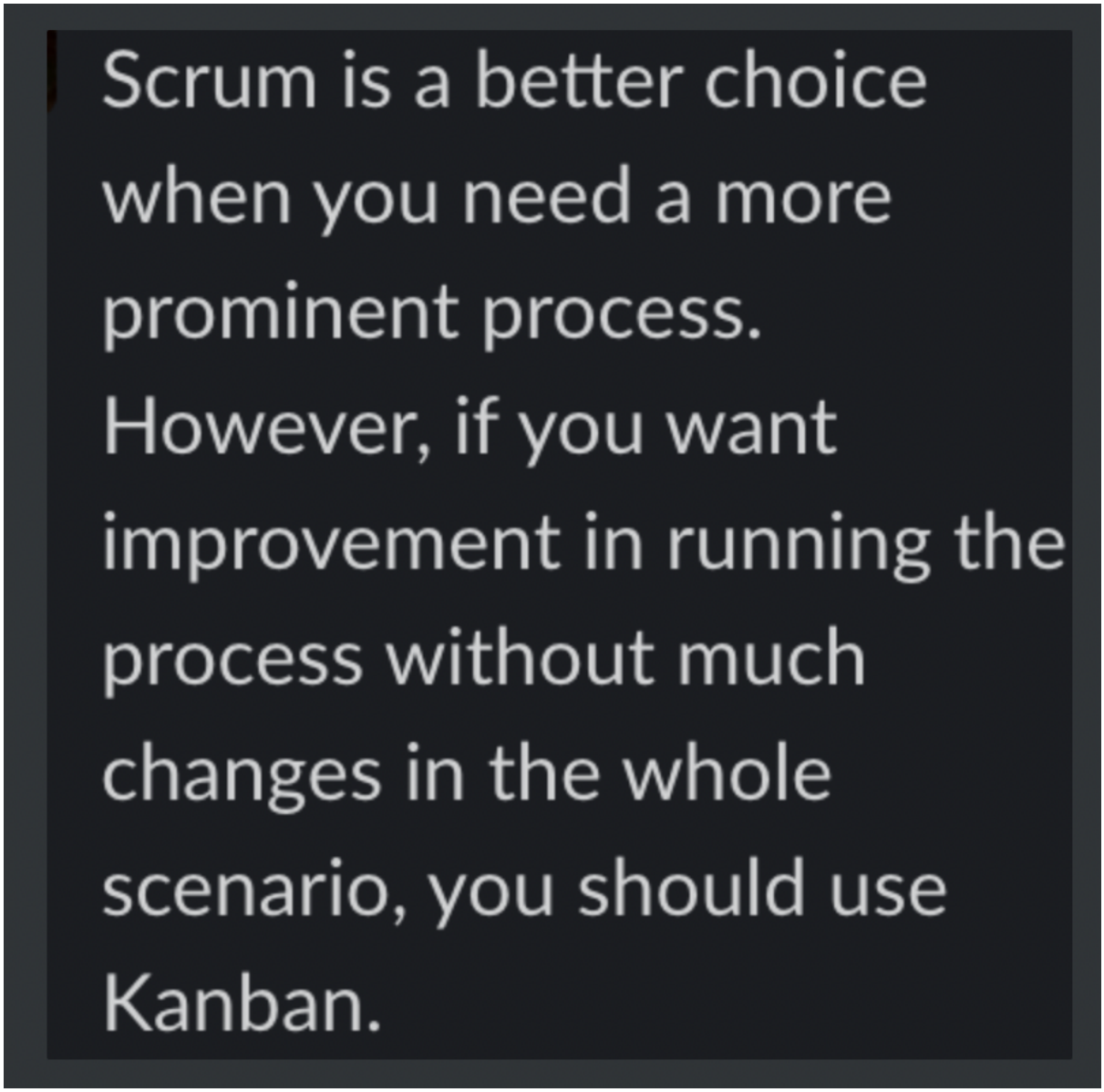
Principles of Kanban

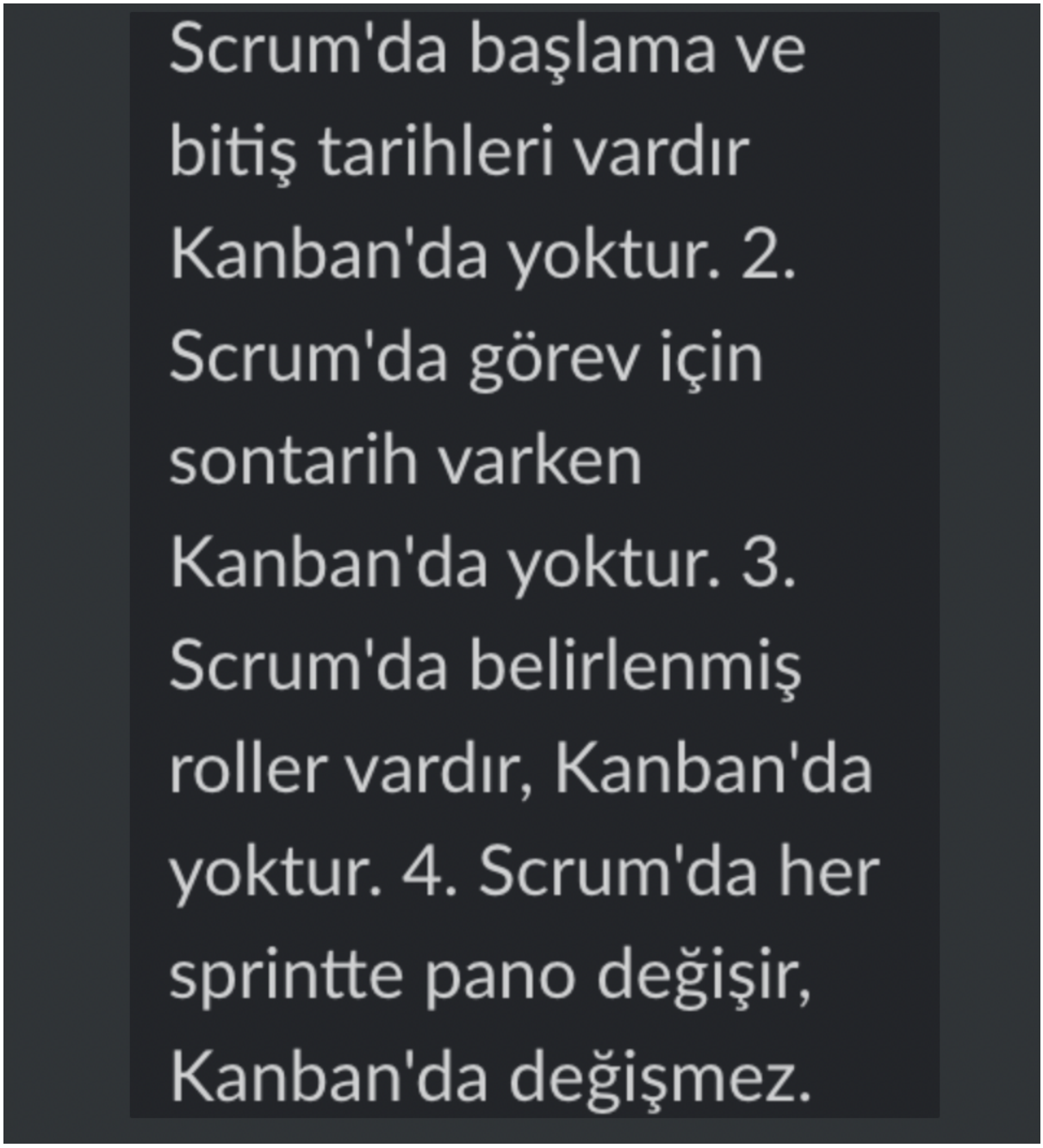
Kanban has adopted **Four Foundational Principles** and **Six Core Practice** to manage the workflow and increase productivity. The four principles of kanban are:

Principlesof Kanban


In kanban, each stage of a project is demonstrated with a what?
card
board
column

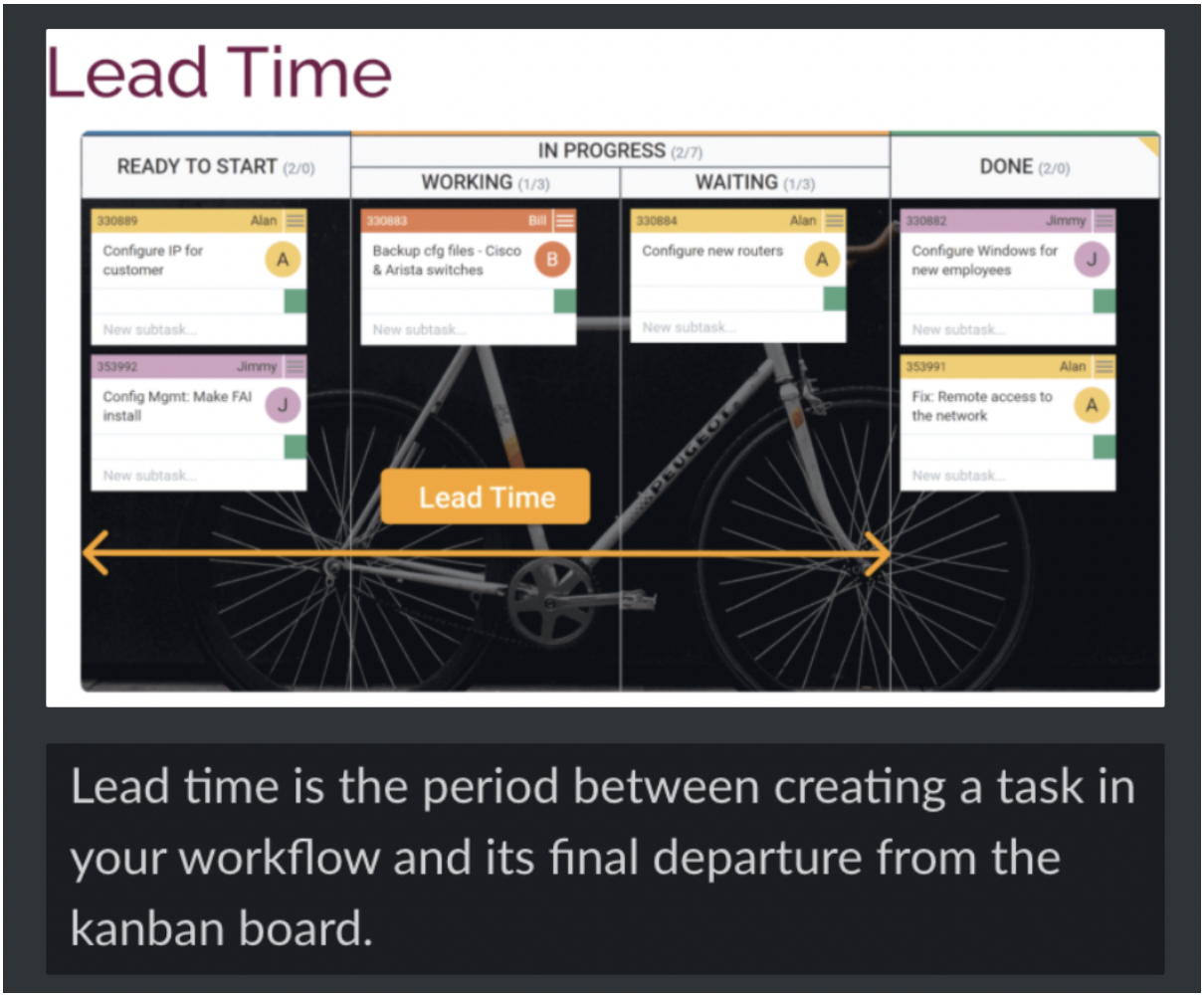

Backlog
Design
4
Develop
2
Test
6
Done
WIP Limits


Scrum is a better choice when you need a more prominent process. However, if you want improvement in running the process without much changes in the whole scenario, you should use Kanban.


Scrum'da başlama ve bitiş tarihleri vardır Kanban'da yoktur. 2. Scrum'da görev için sontarih varken Kanban'da yoktur. 3. Scrum'da belirlenmiş roller vardır, Kanban'da yoktur. 4. Scrum'da her sprintte pano değişir, Kanban'da değişmez.

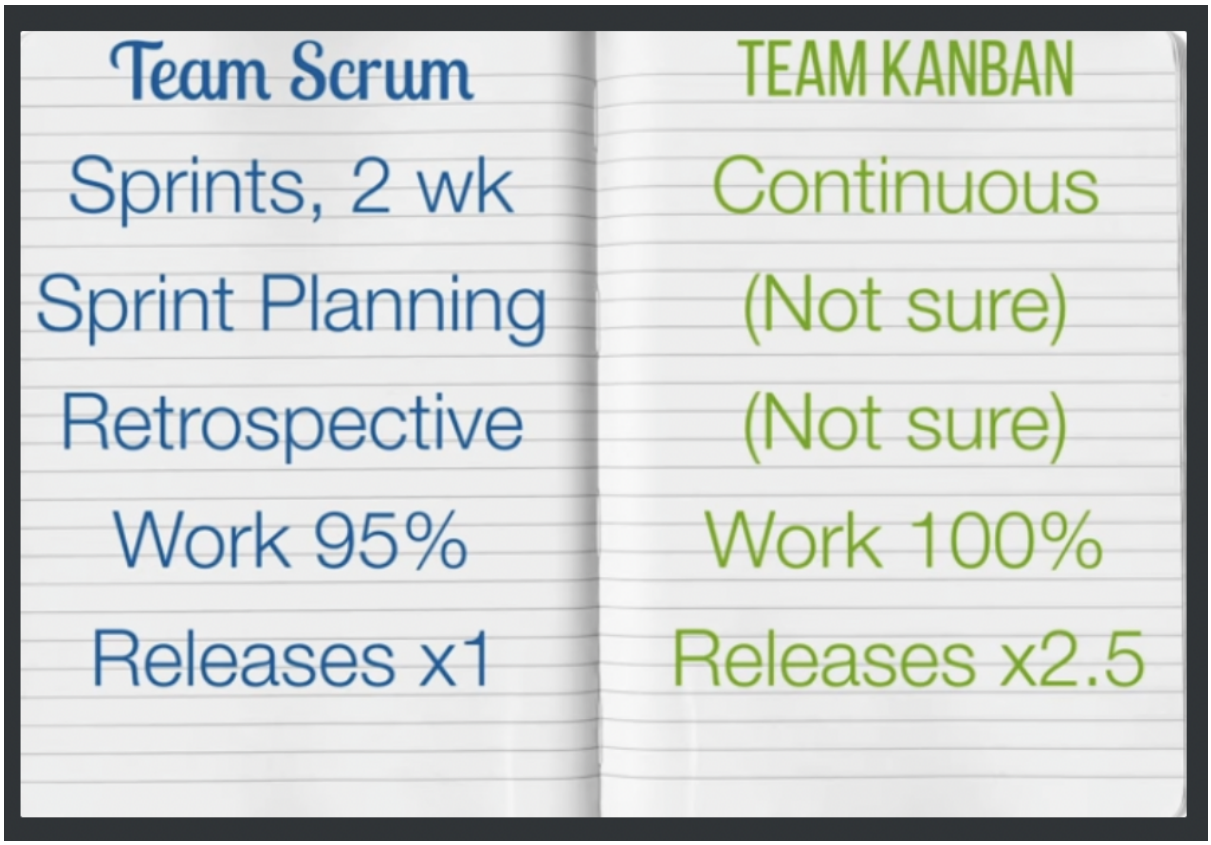
In order to compare the effectiveness of Kanban versus Scrum on software projects, a survey was designed to include various questions about the company, software projects, project management methodology, implementation, and project feedback. Mean, standard deviation, and correlation results were computed to compare the effects that Scrum and Kanban have on the factors. While the results imply that there is no statistically significant difference at the 95% confidence level between Kanban and Scrum for the factors, the results do suggest that Kanban performs better than Scrum in terms of managing project schedule (i.e. the schedule factor). Results also suggest that projects using the Kanban methodology can experience greater consistency in terms of the project management factors. Results also suggest that overall, both Scrum and Kanban lead to successful software projects, where on average, the survey respondents responded with the "Agree" response to questions pertaining to the quality factor. Companies should be aware of the differences in the practical implementation of these methodologies, and choose one or the other based on the context, practical needs, and resources of the project.

In the future, we plan to consider the impact of additional non-quantitative techniques, such as team commitment, work organization, schedule managing, allocation of resources, and visibility. We will also consider collecting additional survey responses to questions pertaining to all project management factors.



> **Q**: Ideally, how WIP limit is calculated with respect to team size?
> **A**: You can start with a WIP limit of 1 to 1.5 times the number of people taking part in each stage or each column. For example, if team size is 4 in a particular stage, max 6 items can be in progress at any given time.

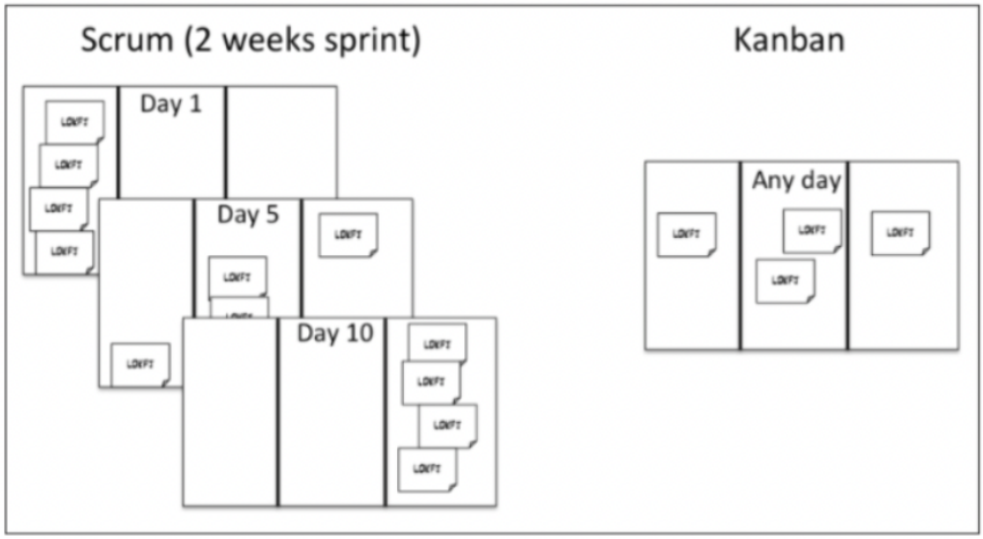
SixCorePracticesofKanban



Comparing Kanban with Scrum

SCRUM	KANBAN
Work is done within time-boxed sprints, generally of 2-4 weeks. The goal is to produce a potentially shippable product after each sprint.	There are no fixed-length sprints. Instead teams pull tasks from a prioritized backlog of things that need to be done.
Product is released on a particular cadence, which is determined by the sprints' length. So a team may release after 3 sprints, or every 6 weeks.	Releases occur continuously, or whenever there is a shippable product created.
There's a heavy focus on cross-functionality. Teams have no specified roles; everyone is a "marketer."	Team members can specialize and pull tasks related to their area of expertise, but too much specialization will reduce a team's effectiveness.
Sprint kickoffs, daily standups, sprint reviews, and sprint retrospectives are vital rituals within the Scrum process.	There is an emphasis on continually improving processes, but no standardized regular meetings or rituals.

Comparing Kanban with Scrum

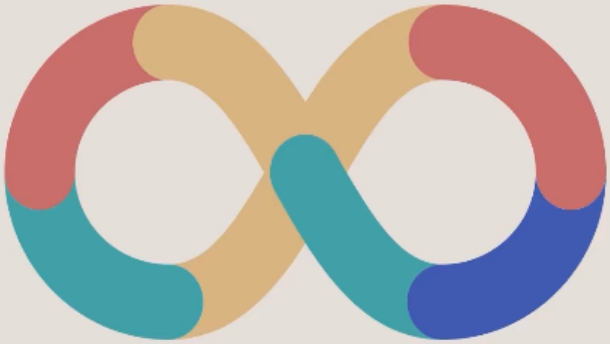


In Scrum you work in iterations called Sprints. In Kanban there is more of a constant flow

DevOps Origin

Origin of DevOps

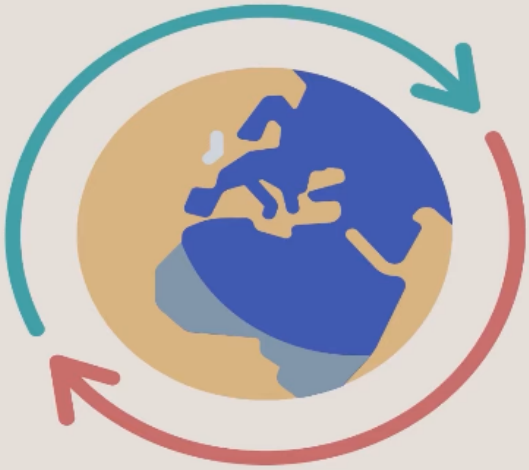
- Waterfall, Agile and lean are the major methodologies used In software development
- Each model has its own pros and cons
- To overcome the drawbacks of these models DevOps methodology is originated
- Its main feature is team coordination and automation



Origin of DevOps

DevOps Trivia

- The concept of **DevOps** emerged out of a discussion between Andrew Clay and Patrick Debois in 2008
- They were concerned about the drawbacks of Agile and wanted to produce something better
- The idea slowly began to spread and after the DevOpsDays event held in Belgium in 2009, it became quite a buzzword
- DevOps is a combination of 2 words, developer and operations

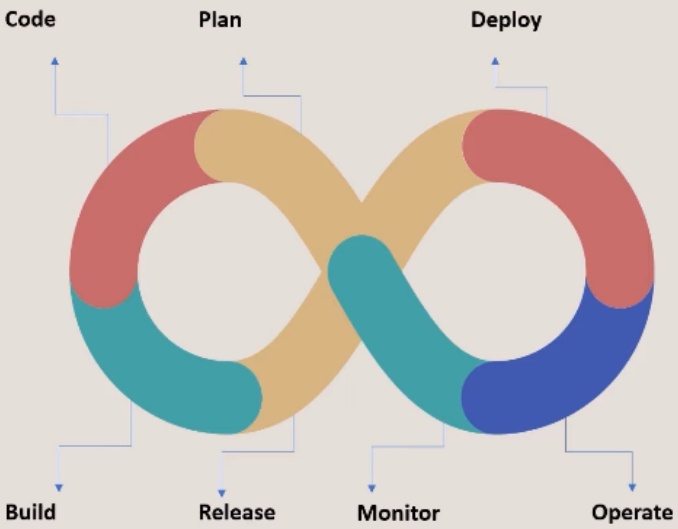


DevOps Trivia

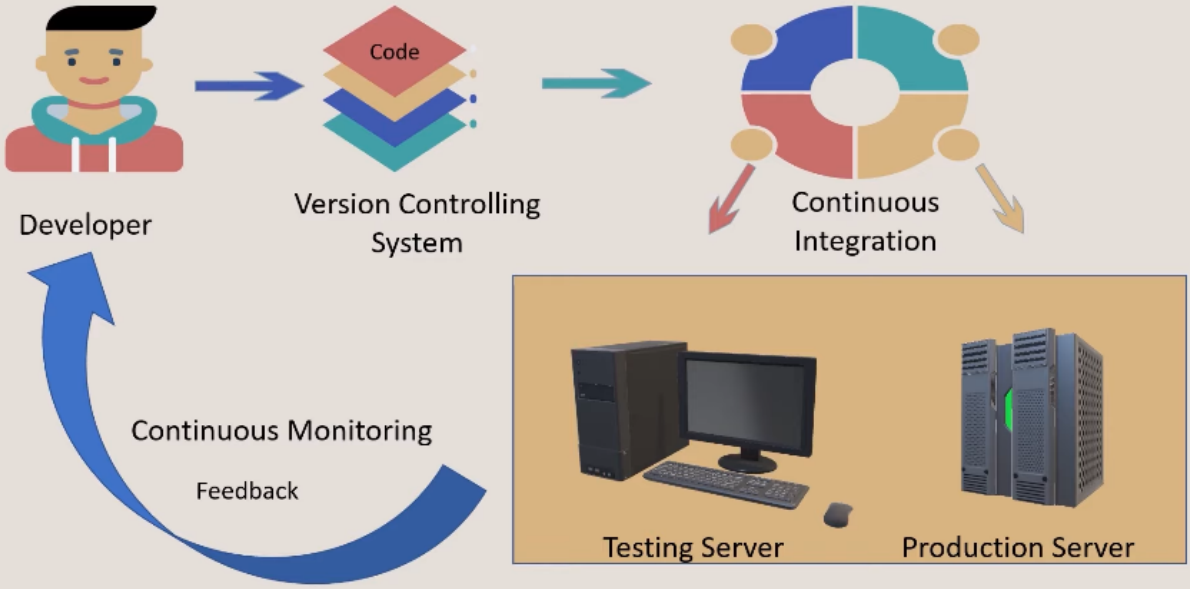
What is Devops

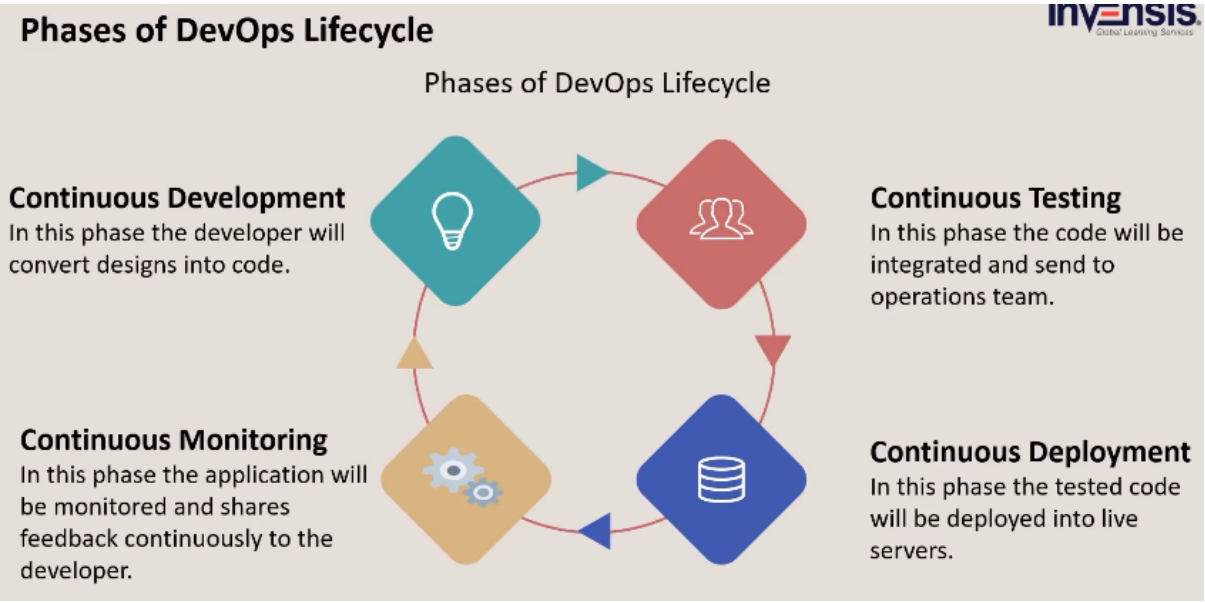
Introduction to DevOps

- DevOps is a methodology which helps in collaboration between Developer and Operations team
- DevOps follows this cooperative approach across teams will help to deliver the project on time
- DevOps is used for automation of tasks in Software Development Life Cycle
- Automation in DevOps Life cycle, the maintenance of software become easy

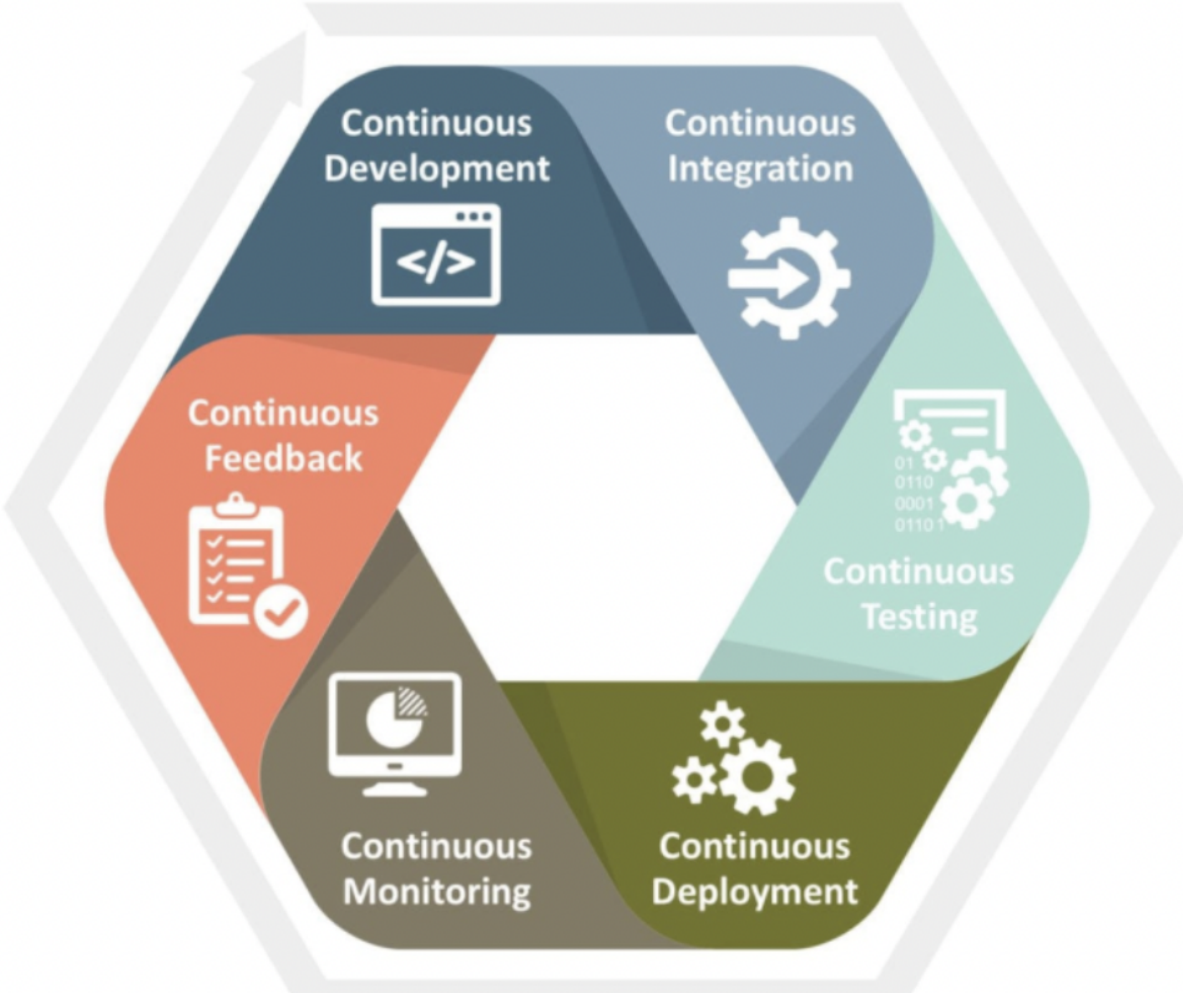


Pictorial Representation of DevOps Lifecycle



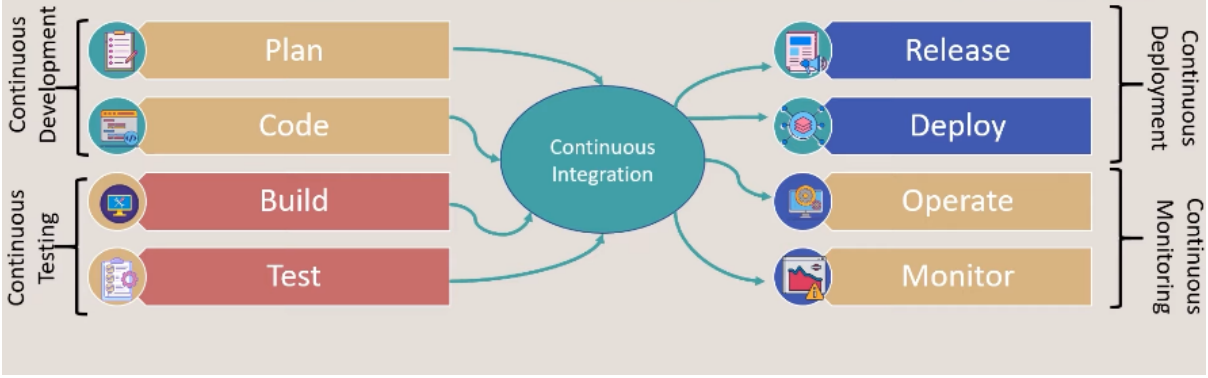


DevOps Model consists of various stages such as continuous development, continuous integration, continuous testing, continuous deployment, continuous monitoring, and continuous feedback.

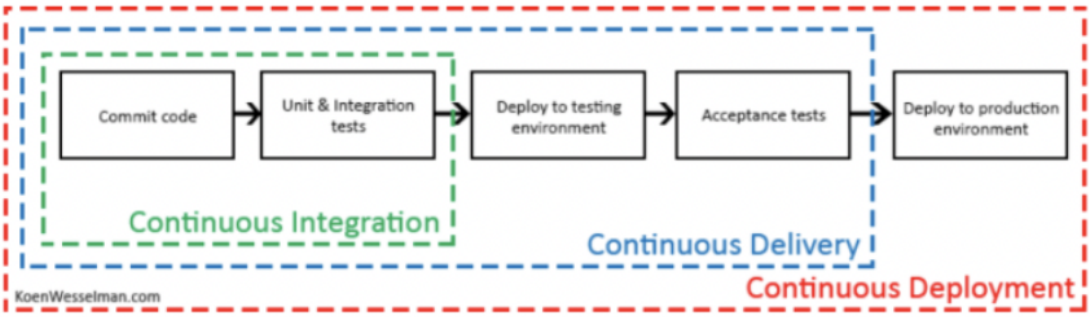


Phases of DevOps Lifecycle (Continued)...

Continuous Development	Continuous Testing	Continuous Deployment	Continuous Monitoring	Continuous Integration
In this phase the development of application will takes place.	In this phase the application code will be tested by ops team.	In this phase the code will be released for end users.	In this phase the released code will be monitored for its maintenance.	This phase is responsible for automating all other phases.



Automating an Integrated DevOps System



Traditional way

With CI &CD

Continuous Development

This is the phase that involves **planning** and **coding** of the software. The vision of the project is decided during the planning phase and the developers begin developing the code for the application. There are no DevOps tools that are required for planning, but there are a number of tools for maintaining the code.

Continuous Testing

This is the stage where the developed software is continuously tested for bugs. For Continuous Testing, automation testing tools like Selenium, TestNG, JUnit, etc are used.

Continuous Integration

This stage is the **heart** of the entire DevOps life cycle. It is a software development practice in which the developers require to commit changes to the source code more frequently.

This may be on a daily or a weekly basis. Every commit is then built and this allows early detection of problems if they are present. Building code not only involves compilation but it also includes code review, unit testing, integration testing, and packaging.

The code supporting new functionality is continuously integrated with the existing code. Since there is continuous development of software, the updated code needs to be integrated continuously as well as smoothly with the systems to reflect changes to the end-users.

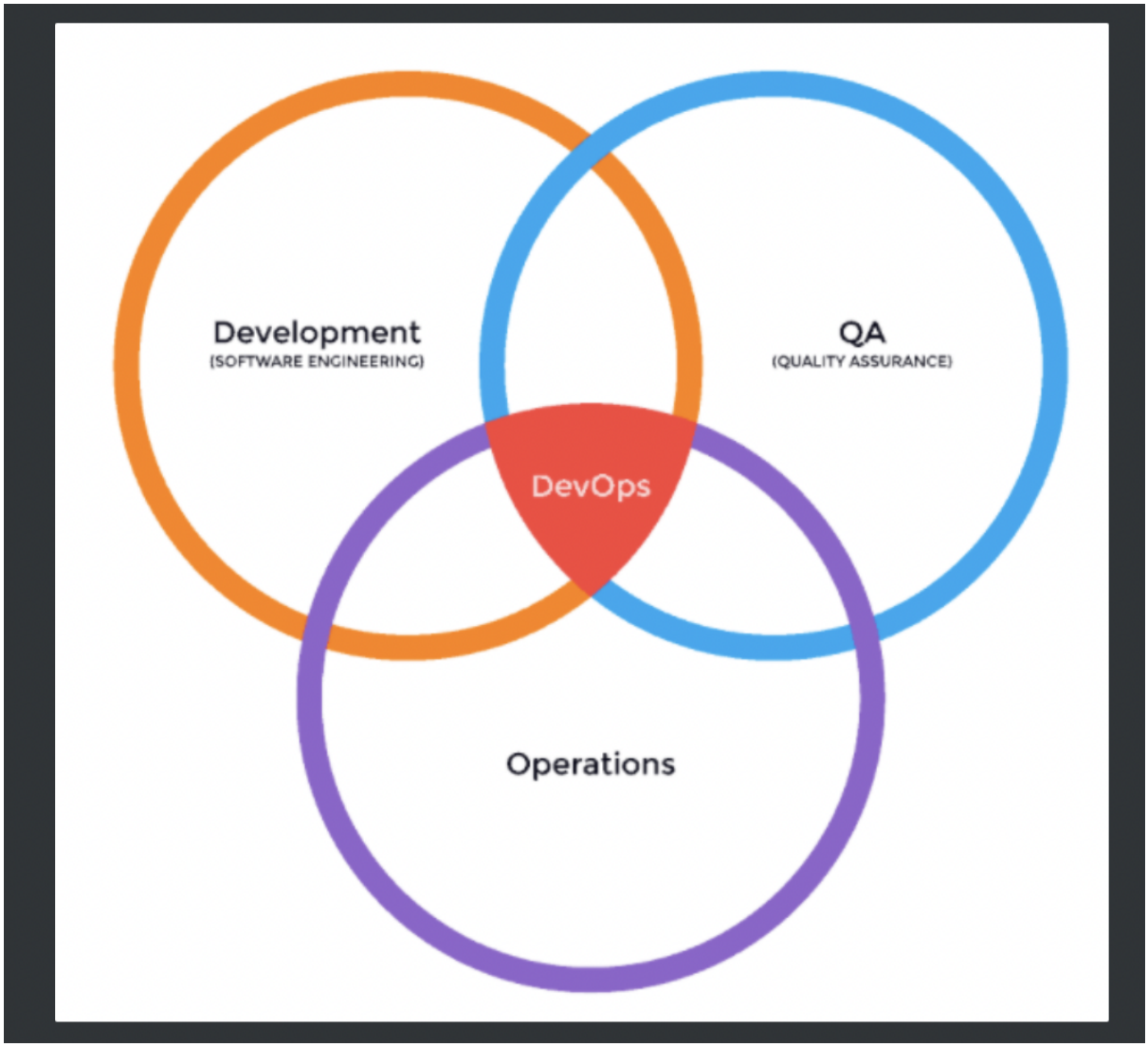
Jenkins is a very popular tool used in this phase.

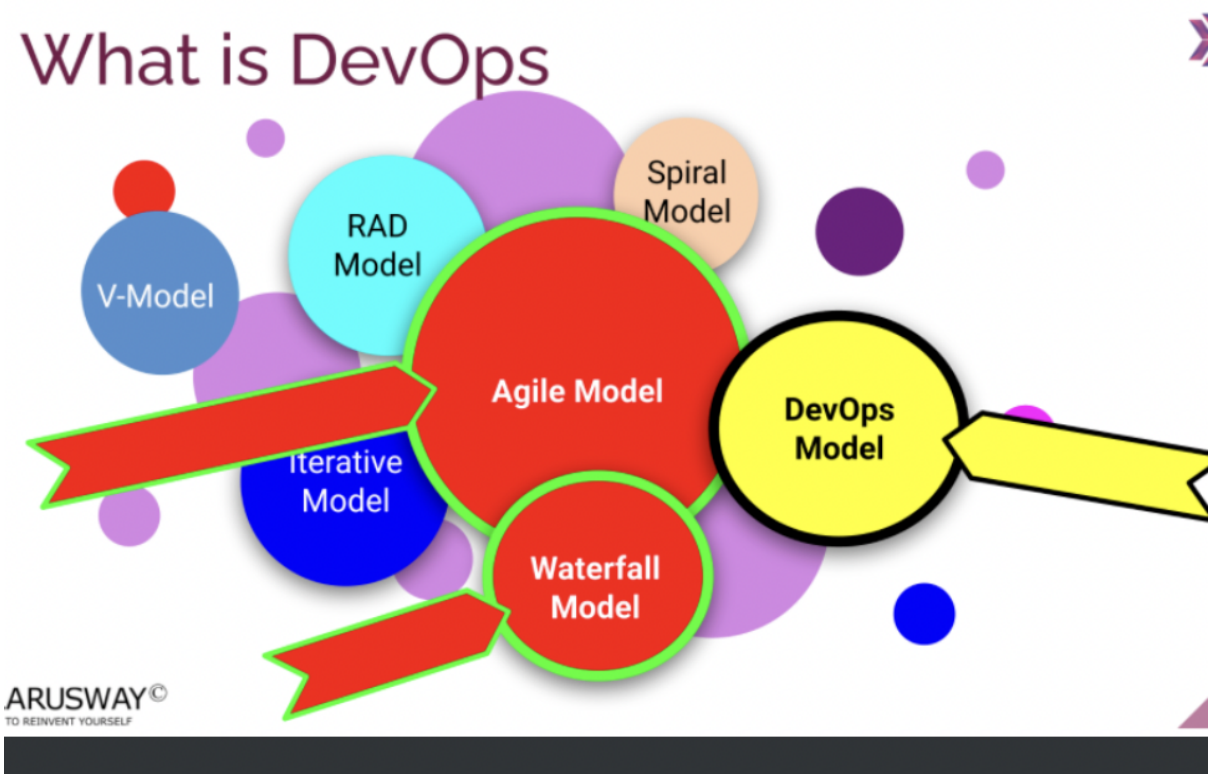
What is DevOps

The term DevOps is the short form of **Development** and **Operations**.

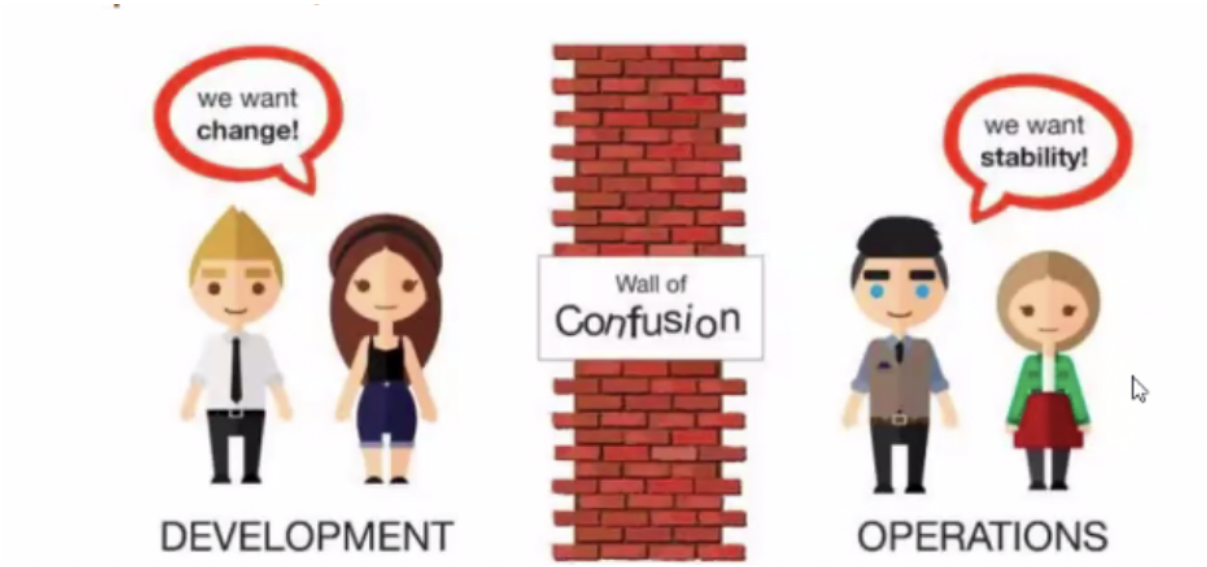
It focuses on collaboration between developers and other roles.

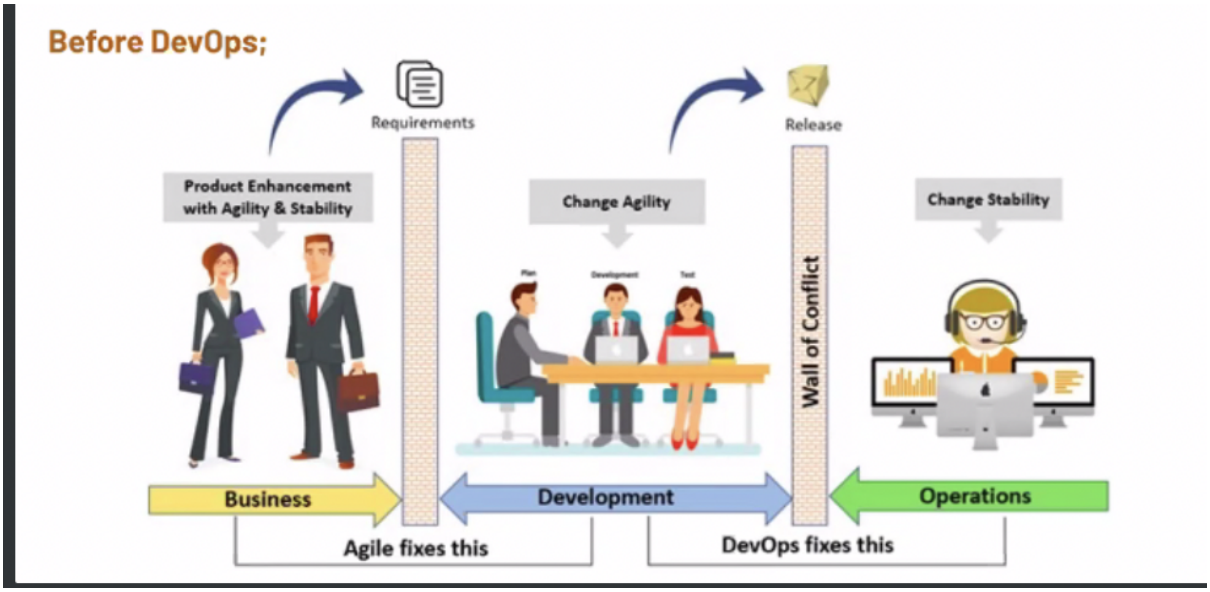
DevOps is a practice that allows a single team to manage the entire application development life cycle, that is, development, testing, deployment, operations.

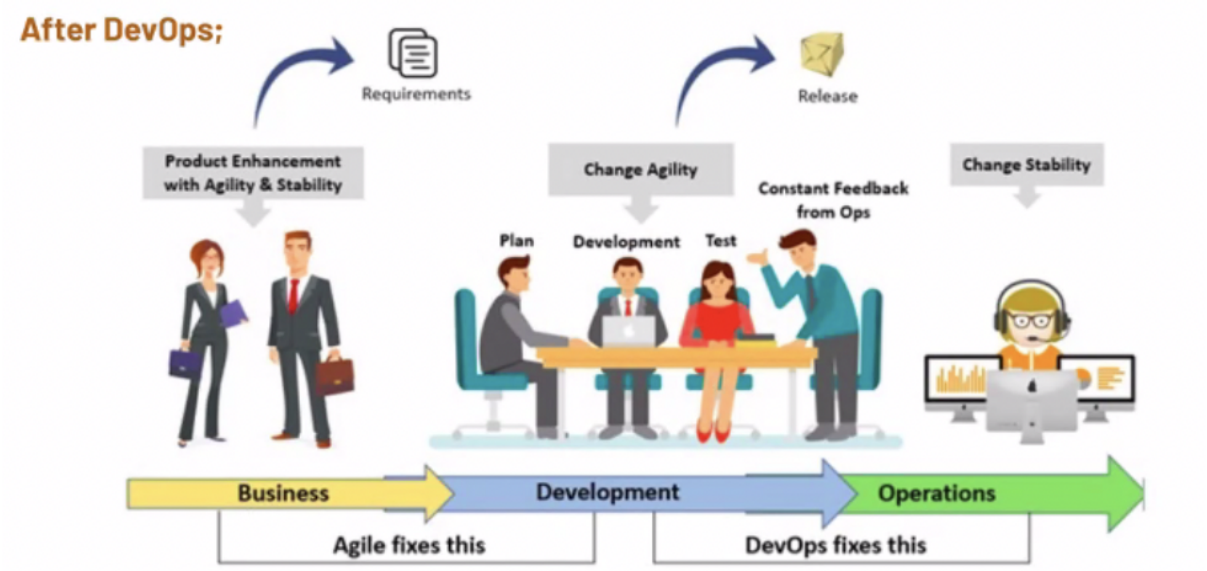

Development
(SOFTWARE ENGINEERING)
QA
(QUALITY ASSURANCE)
DevOps
Operations

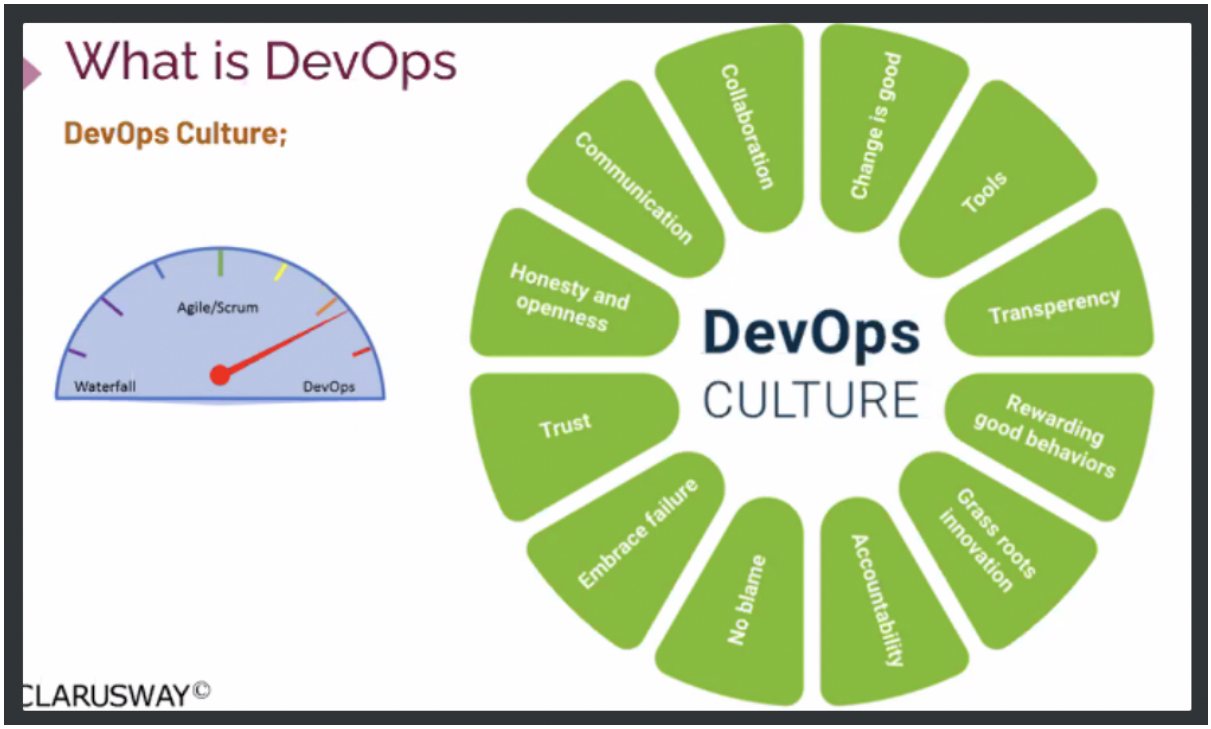


- **DevOps** is a set of practices that combines software **development** (Dev) and IT **operations** (Ops).
- It aims to **shorten** the systems development life cycle and provide **continuous delivery** with **high software quality.**
- DevOps is complementary with Agile software development; several DevOps aspects came from the Agile methodology.
- DevOps addressed the **gap** between Developers and Operations.


we want change!
Wall of Confusion
we want stability!
DEVELOPMENT
OPERATIONS

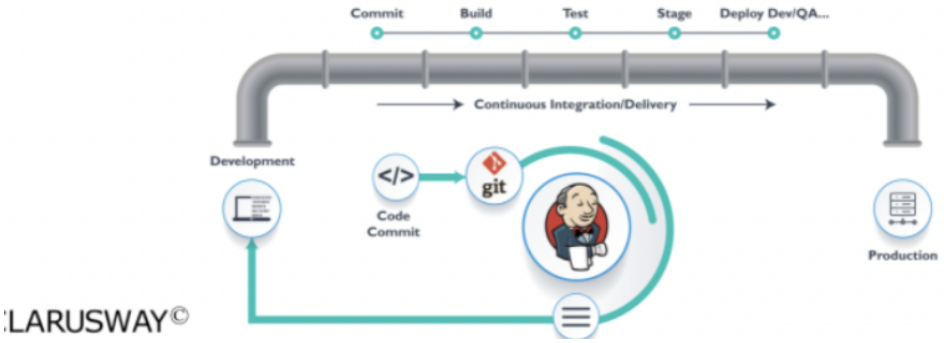

Before DevOps;
Requirements
Release
Product Enhancement with Agility & Stability
Change Agility
Change Stability
Wall of Conflict
Business
Development
Operations
Agile fixes this
DevOps fixes this


After DevOps;
Requirements
Release
Product Enhancement with Agility & Stability
Change Agility
Constant Feedback from Ops
Change Stability
Plan
Development
Test
Business
Development
Operations
Agile fixes this
DevOps fixes this



Pipelines;

- A **Pipeline** is a **chain** of **tasks** that can be **automated**.
- Integration tools use pipelines to perform tasks repetitively and continuously
- The process is called Continuous Integration (CI)
- Pipelines keep work flowing forward in our DevOps system



What is Jira?

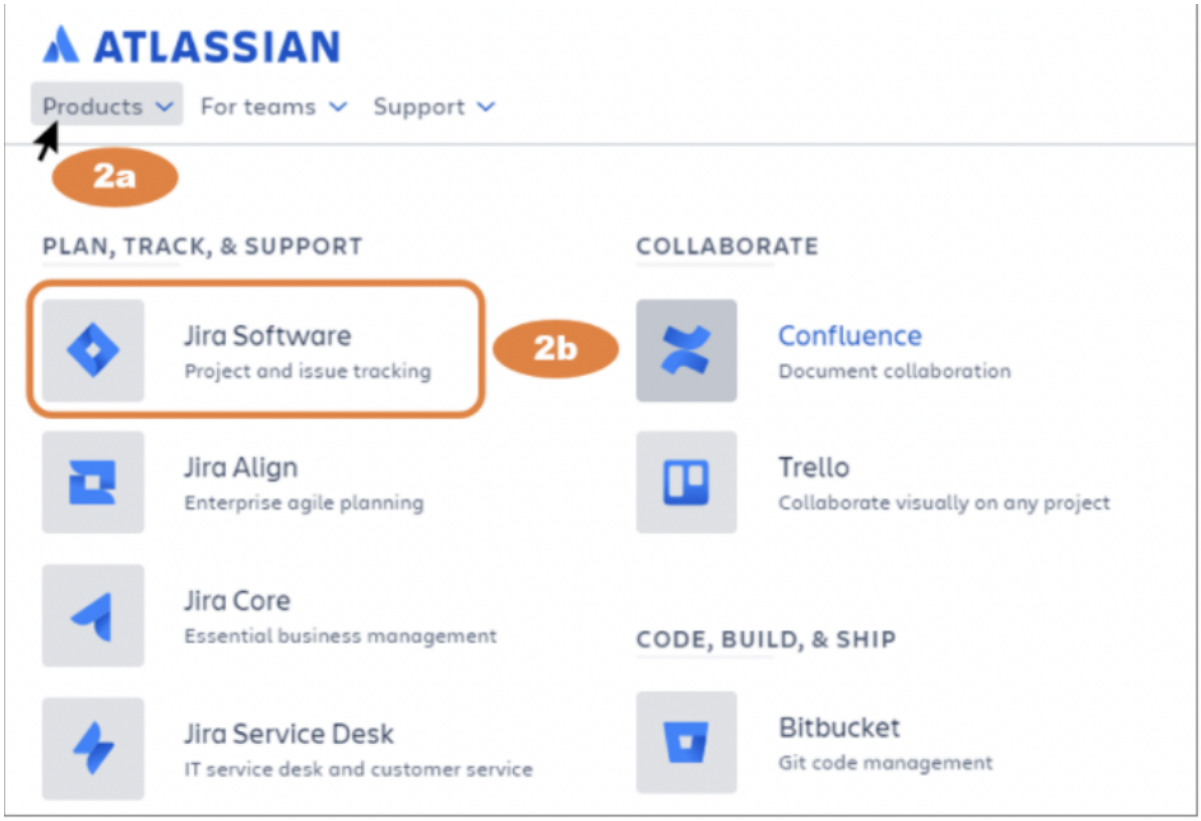
Jira is a tool developed to help teams for project management, bug tracking, and issue tracking. In simple terms, it is an **issue tracker**. Jira is widely used by big companies in software development and software testing. It is web-based and licensed product created by Australian Company Atlassian.

- A tool used to help teams perform, visualize and manage work.
- Models the team's current processes/workflows.
- You can open tickets or issues and push them through various statuses in your processes

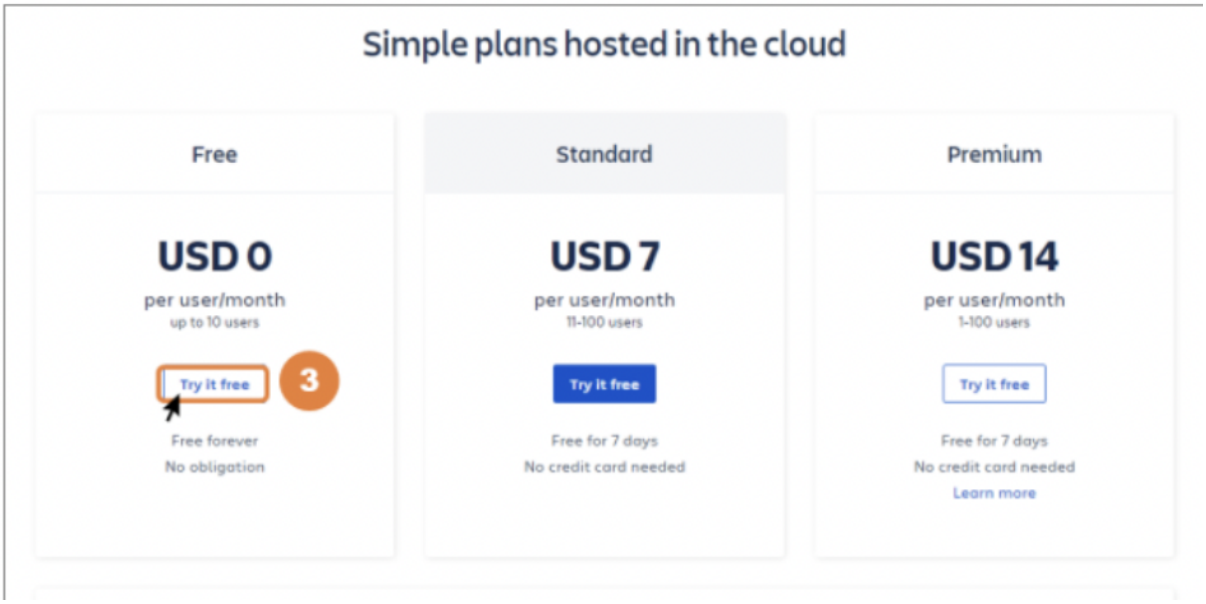
Why Jira?

- Leverage project management technology, allowing teams to focus on their work.
- Facilitates planning, prioritizing, organizing and completing work.
- Visualizes work using project boards, reports and dashboards.
- Facilitates team communication.

CLARUSWAY©



Product	Description
Jira Software	This is a software development tool used by agile teams (formerly known as JIRA Agile). It is very suitable for software development teams who want to use agile methodologies such as Scrum and Kanban.
Jira Align	Jira Align is a member of the Jira family which allows the company leadership to monitor current developments and link them to business outcomes at an enterprise scale.
Jira Core	This is similar to the classic Jira, optimized for business teams, with all the field customizations and workflow capabilities. Jira Core keeps the teams organized by managing projects, monitoring details, and measuring performance.
Jira Service Desk	Jira Service Desk is a modern IT service desk software that can run on the cloud or server. With this product, service desks can easily receive, monitor, manage and resolve the requests from customers.



The name "Jira" is derived from the truncation of *Gojira*, the Japanese word for *Godzilla*. Here, a reference is made to the competitor *Bugzilla*.

Jira Core	Managing Job Applicants in Human Resources ✔
Jira Software	Creating a Hotel Booking iOS App ✔
Jira Service Desk	Helping Customers with A Help Desk ✔

💡**Tips:**

- The **Jira Administrator** controls all access to Jira Applications and what users can do in Jira.

Jira Software Hosting Options

Cloud

With Jira Software Cloud, the Jira Software site is hosted and installed in the cloud for you. This hosting option is usually preferred by the teams wh want to get started quickly and practically. Thus, agile teams are free from managing the technical complexity of the hosting process.

A team chooses the cloud option for a variety of reasons, as shown below:

Which one is named as collection of epics that drive toward a common goal?

- Theme
- Initiative
- Story
- Task

Initiatives are made up of Epics

- In multiple quarters to a year

Epics are made up of Stories

- In a month or a quarter

Stories are made up of Subtasks

- In days or a sprint

..... are large bodies of work that can be broken down into a number of smaller stories.

- Epics
- Stories
- Tasks
- Subtasks

Initiative	Collections of epics that drive toward a common goal
Epic	Big chunk of work that has one common objective
Story	Short requirements written from the perspective of an end user.
Subtask	Smaller piece of story

What is Board?

- A board displays issues from one or more projects in columns that represent the team's process.
- There are three types of boards in the Jira Software.

Type of Board	For Which Teams?
Scrum board	Teams that plan their work in sprints.
Kanban board	Teams that focus on managing their WIP.
Next-gen board	Teams who are new to agile.

Project Key

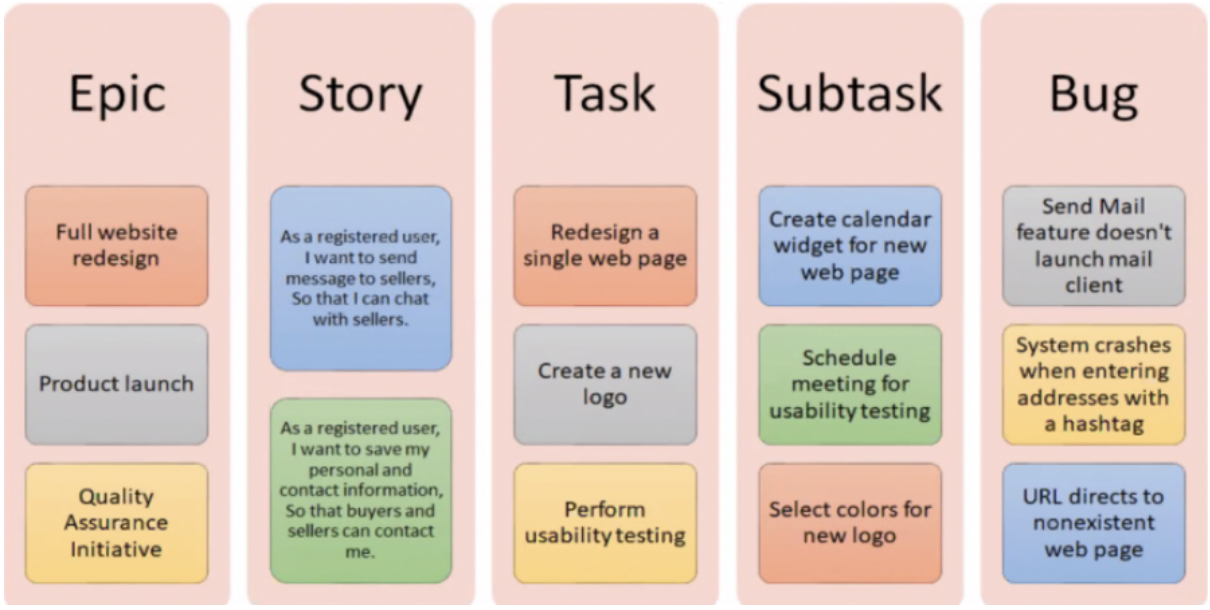
A project key is a unique code for your project.

Jira Software will automatically generate a short project key in accordance with your project name.

However, if you want to specify this auto-generated key yourself, you can change it.

Issue Types

- **Epic** : An epic is a set of jobs that can be divided into manageable user stories.
- **Story** : A story is a short requirement written from an end-user perspective.
- **Task** : A task represents the job that needs to be done.
- **Subtask**: A subtask can be considered as a smaller piece of a story.
- **Bug** : A bug impairs expected functionality of the product.



💡Tips:

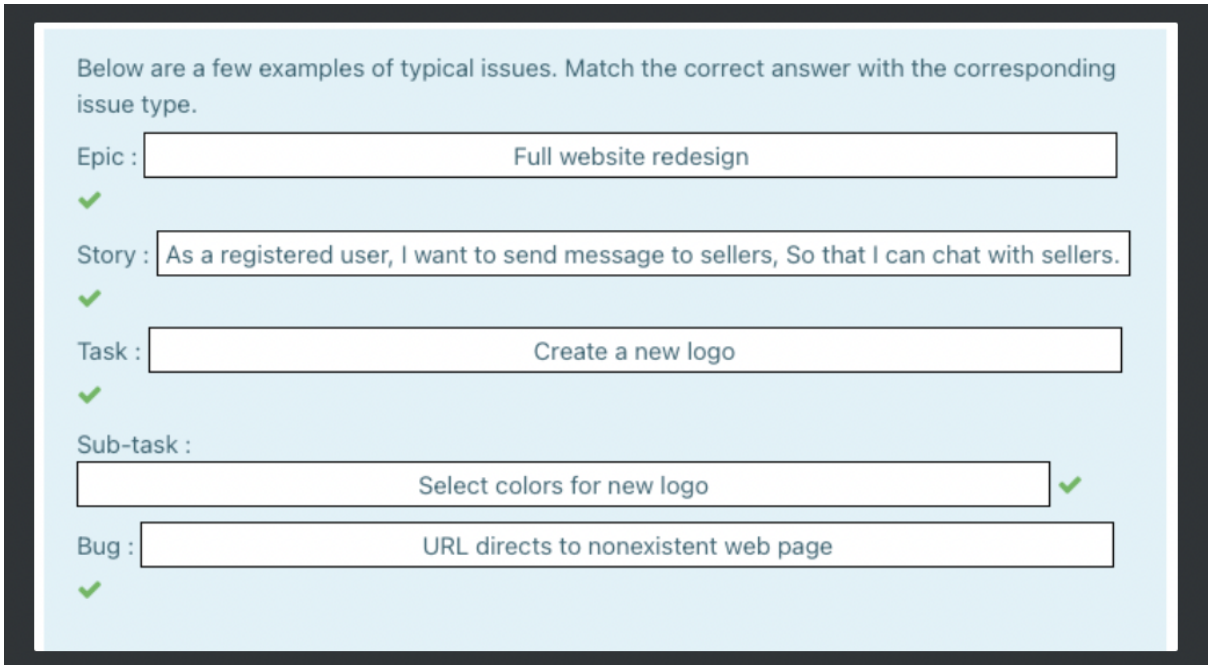
- The hierarchy for units of work in Jira Software is as follows: Project > Epics/Components > Stories > Tasks > Subtasks
- Teams can customize the names of issue types to match their specific project requirements.
- Each issue type can have different fields, screens and workflows.

> **Q**: List and briefly describe the issues types in Jira Software.
> **A**: Issue types can be defined in short sentences as follows: A bug is a problem which impairs or prevents the functions of a product. An epic is a big user story that needs to be broken down. A subtask is a piece of work that is required to complete a task. A user story is the smallest unit of work that needs to be done. A task represents work that needs to be done.

Examples for Issue Types

Here are the icons and some examples for each issue type:

Issue Type	Icon	Examples
Epic		Full website redesign, Product launch, Quality Assurance Initiative
Story		As a registered user, I want to send message to sellers, So that I can chat with sellers.
Task		Redesign a single web page, Create a new logo, Perform usability testing
Sub-task		Create calendar widget for new web page, Schedule meeting for usability testing, Select colors for new logo
Bug		Send Mail feature doesn't launch mail client, System crashes when entering addresses with a hashtag, URL directs to nonexistent web page



Issue Priorities

The priority of an issue determines how important it is relative to other issues. So you can determine which issues your team should focus on first. The default priorities are listed below, in order from highest to lowest.

Priority	Icon	Description
Highest	↑	This problem will block progress.
High	↑	Serious problem that could block progress.
Medium	↑	Has the potential to affect progress.
Low	↓	Minor problem or easily worked around.
Lowest	↓	Trivial problem with little or no impact on progress.

💡**Tips:**

- Keep in mind that both the priorities and their meanings can be customized by the administrator.

> **Q**: What is Jira Workflow?
> **A**: A Jira workflow is a set of statuses and transitions that an issue moves through during its lifecycle, and typically represents a process within your organization. Workflows can be associated with particular projects and, optionally, specific issue types by using a workflow scheme.

Definition of Done

D O D

A contract or agreement among team members that defines what “done” means

Definition of Done

Example DOD

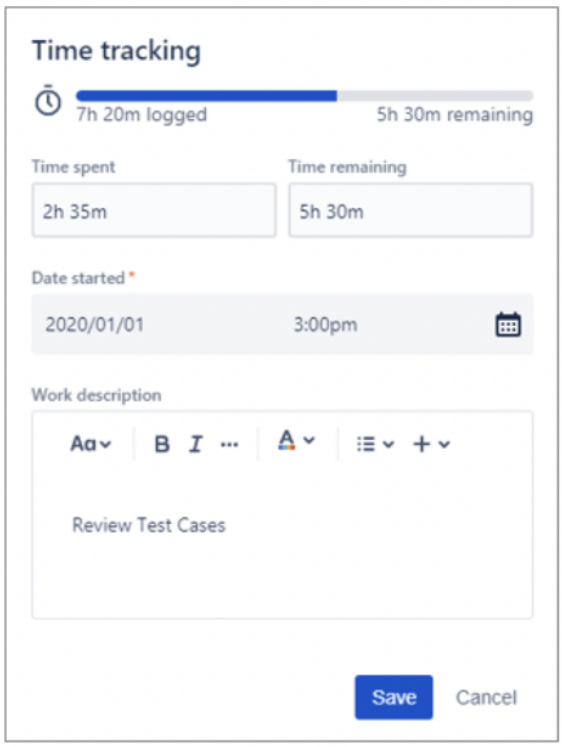
- ✓ Code builds without warnings
- ✓ Code is unit tested
- ✓ Documentation is updated

💡Tips:

- If you find that a subtask is large enough at an issue level, you can convert it into an issue.
- Likewise, if you see that an issue is really just a subtask of another issue, you can convert it into a subtask.

> **Q**: Explain how a subtask is created in Jira Software.
> **A**: Follow these steps to create a subtask. Click on an issue where you want subtasks to be created. From the dialog box, click on the "Create subtask" button. Add a summary to your subtask and click on "Create" button. Click on a subtask that you have created and you can do the followings: Attach file, Link issue, Link page, Change status, Assign an assignee, Assign an reporter, Edit labels, Determine priority.

2. Fill in the **Log time** fields and select **Save**



When logging time, use the format: 2w 4d 5h 45m.

💡**Tips:**

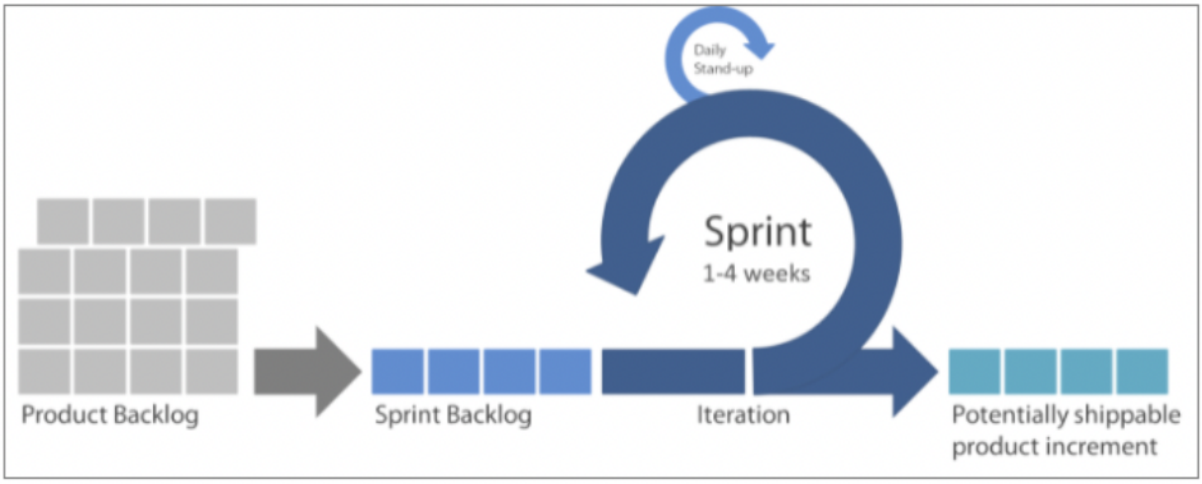
- Time tracking is enabled by default in Jira; if you disable or re-enable, you will not lose any existing data.

Which of the following is a correct format that you need to use for logging time on issues?

Select one:

- ○ 4H 30M
- ◉ 4h 30m ✔
- ○ 4 hours 30 minutes

- Epics are large stories with a distinct start and end, span multiple sprints
- Epics are not groupings of work items
- Epics may contain stories, bugs, and tasks within them
- Stories, bugs, and tasks are all at the same level hierarchically in JIRA
- Stories are called user stories to ensure focus on users

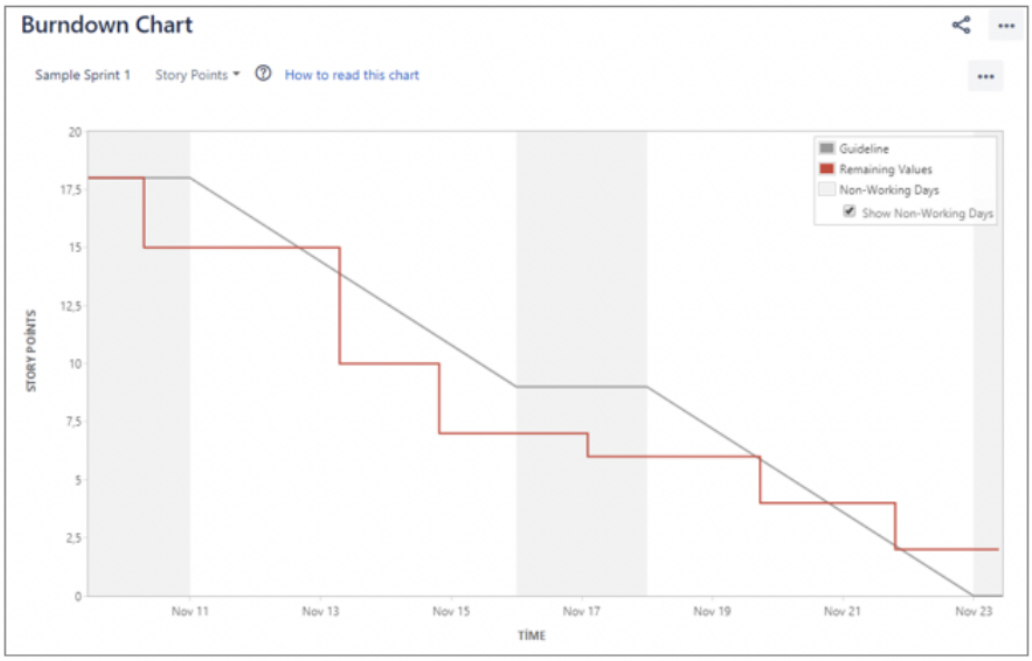


💡**Tips:**

- You can plan multiple sprints at one time.
- You won't be able to start a new sprint until the active sprint is completed unless your Jira administrator has configured parallel sprints.

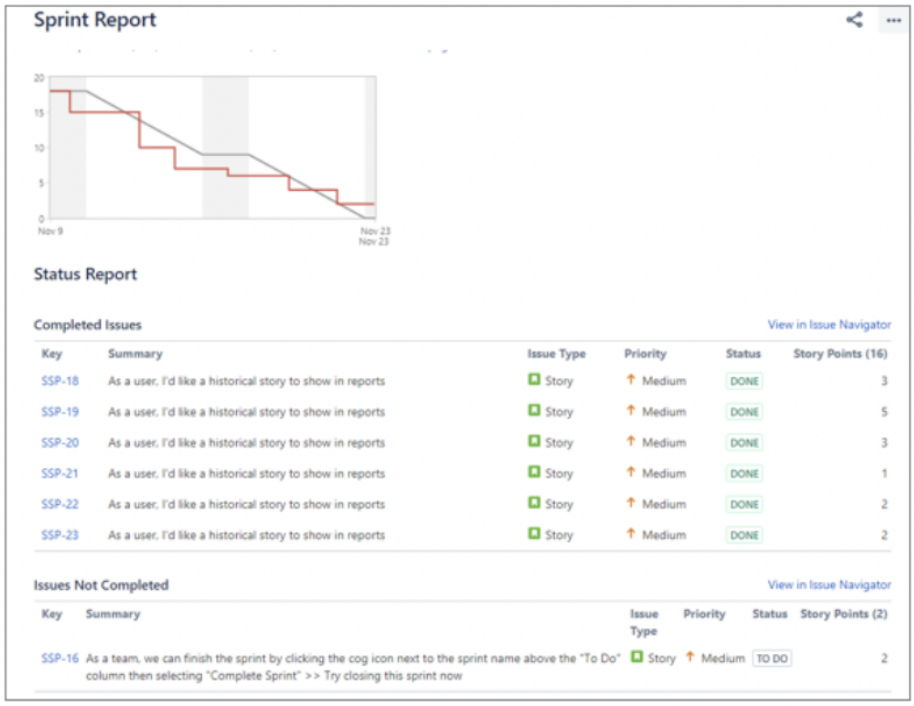
Agile reports will also be useful during sprint retrospective meetings ✔ because they provide detailed information on how sprint is progressing.

A Burndown chart is a graphical representation of the work left to be done (STORY POINTS) versus the time remained in the active sprint (TIME). It illustrates work completed and work remaining to be done. The Burndown Chart gives everyone involved a clear sense of the progress of the project.

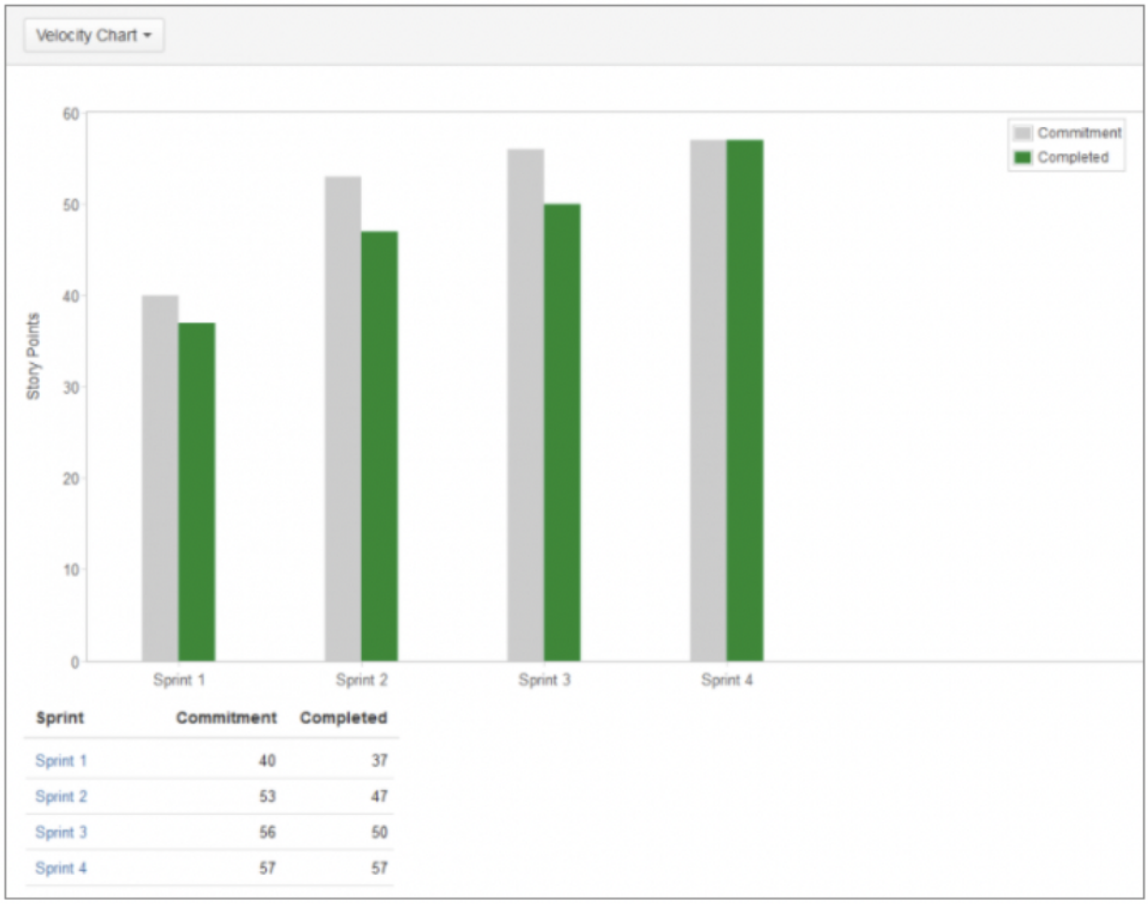


The Sprint report displays a condensed burndown chart and lists which issues are complete and which are still incomplete.

The Sprint report also shows items that were added to the sprint late, thus changing the sprint's scope. These reports are useful for mid-sprint check-ins as well as retrospective meetings.



The Velocity Chart shows the amount of value delivered in each sprint, enabling you to predict the amount of work the team can get done in future sprints. It is useful during your sprint planning meetings, to help you decide how much work you can feasibly commit to.

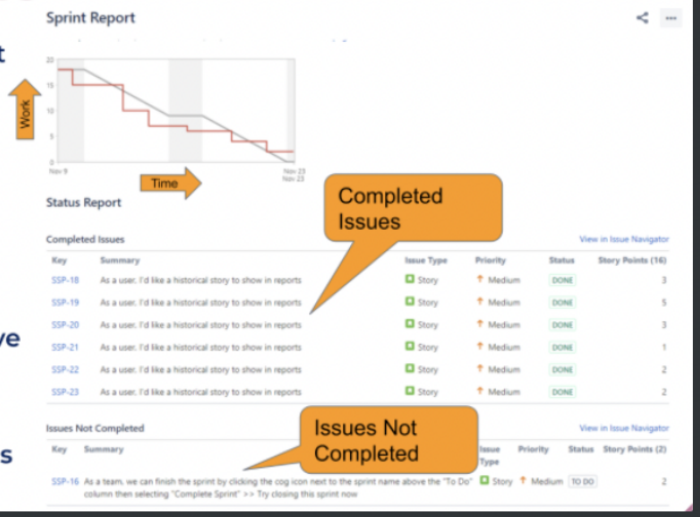


Sprint	Commitment	Completed
Sprint 1	40	37
Sprint 2	53	47
Sprint 3	56	50
Sprint 4	57	57

The y-axis displays the statistic used for estimating stories. In the example above, the team is using story points. The x-axis displays the sprints completed by the team. The gray bar for each sprint shows the total estimate of all issues in the sprint when it begins. The green bar in each sprint shows the total completed estimates when the sprint ends.

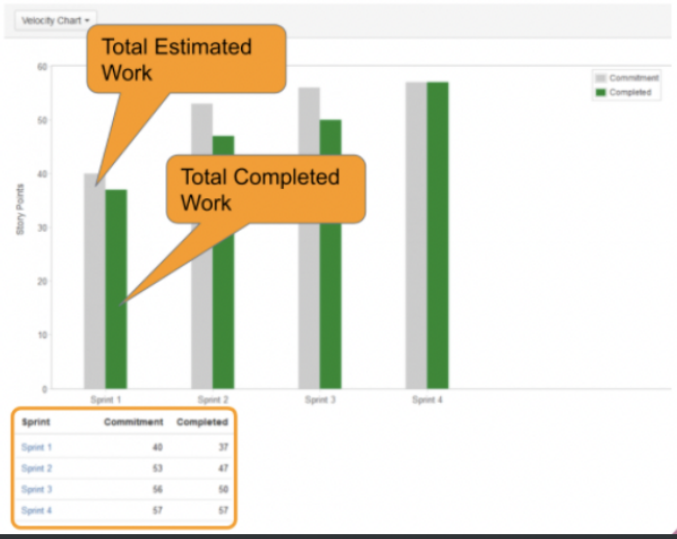
Sprint Reports

- Condensed Burndown Chart
- List of complete and incomplete issues
- Summary of sprint progress
- Useful for sprint retrospective meetings
- Only applies to scrum boards



Velocity Chart

- Shows estimates versus actual work completed
- Helps predict future performance
- Identify trends in commitments & delivery
- Useful for sprint planning to determine scope



Sprint	Commitment	Completed
Sprint 1	40	37
Sprint 2	53	47
Sprint 3	56	50
Sprint 4	57	57

In a Velocity Chart, which bar represents which label? Match them!

Gray bar	Commitment ✔
Green bar	Completed ✔

What is Board?

- A board displays issues from projects in columns that represent the team's process.
- There are two types of boards in the Jira Software.

Type of Board	For Which Teams?
Scrum board	Teams that plan their work in sprints.
Kanban board	Teams that focus on managing their WIP.

Creating Versions

VERSIONS

- Points of a project over time
- New features and fixes

What is an Issue?

In Jira, teams use **issues** to track individual pieces of work that must be completed.

Issue Priorities

The priority of an issue determines how important it is relative to other issues.

Priority	Description
Highest	This problem will block progress.
High	Serious problem that could block progress.
Medium	Has the potential to affect progress.
Low	Minor problem or easily worked around.
Lowest	Trivial problem with little or no impact on progress.

References:

Clarusway

Invensis

study.com