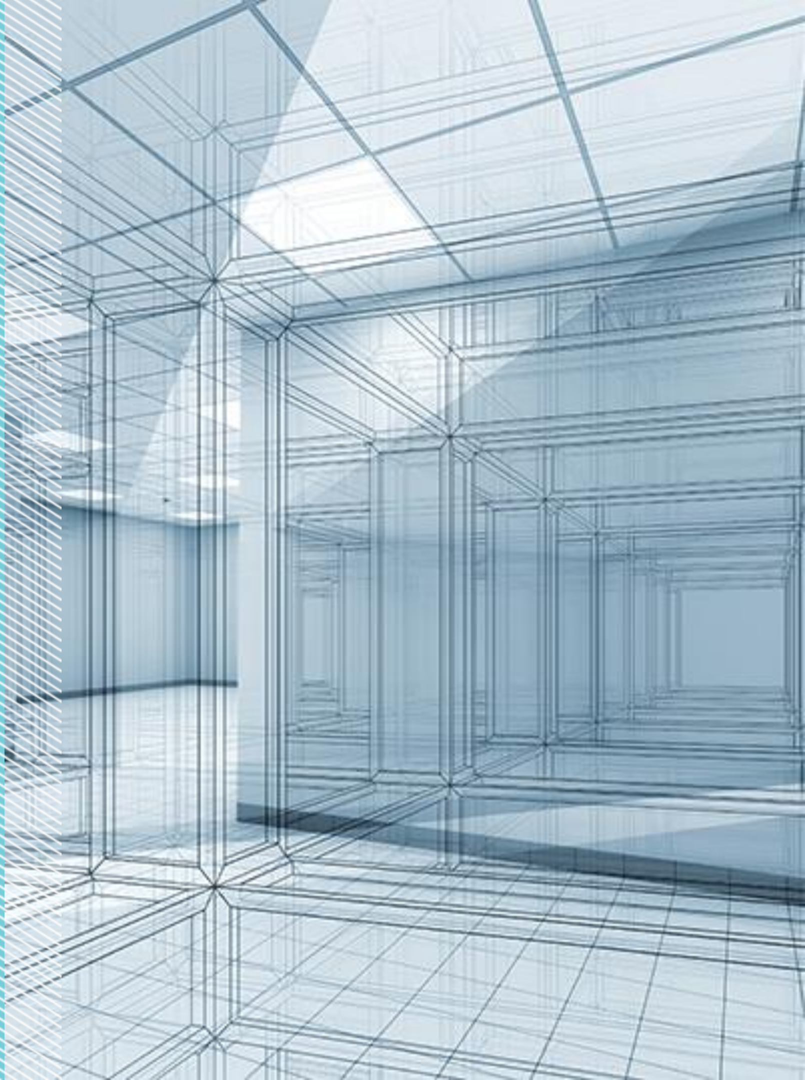
SDLC methodologies

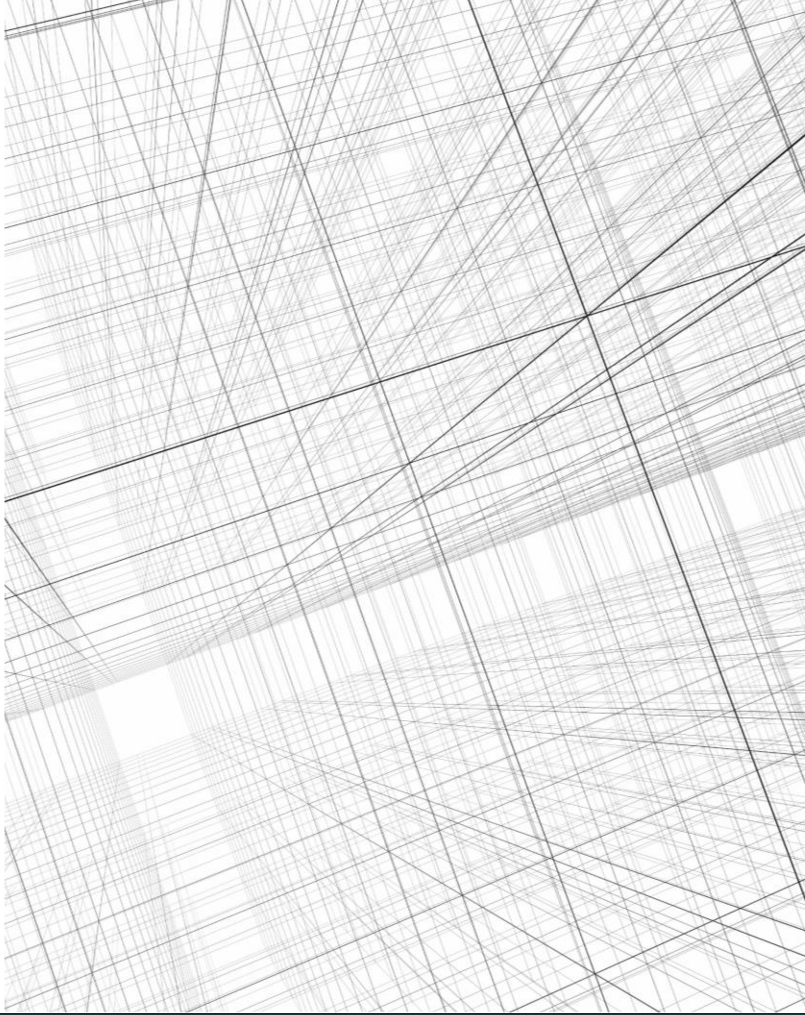
Anna Kuzina, Project manager

July 2022



Course objectives

- SDLC – WHAT IS IT?
- SDLC MODELS – CLASSIC MODELS, INCREMENT & ITERATION
- AGILE – A BIG FAMILY OF PRACTICES, METHODOLOGIES, FRAMEWORKS
- SCRUM – A LIGHTWEIGHT FRAMEWORK THAT HELPS TO GENERATE VALUE
- KANBAN – FROM JAPAN WITH LOVE
- DEVOPS ROLE IN SDLC

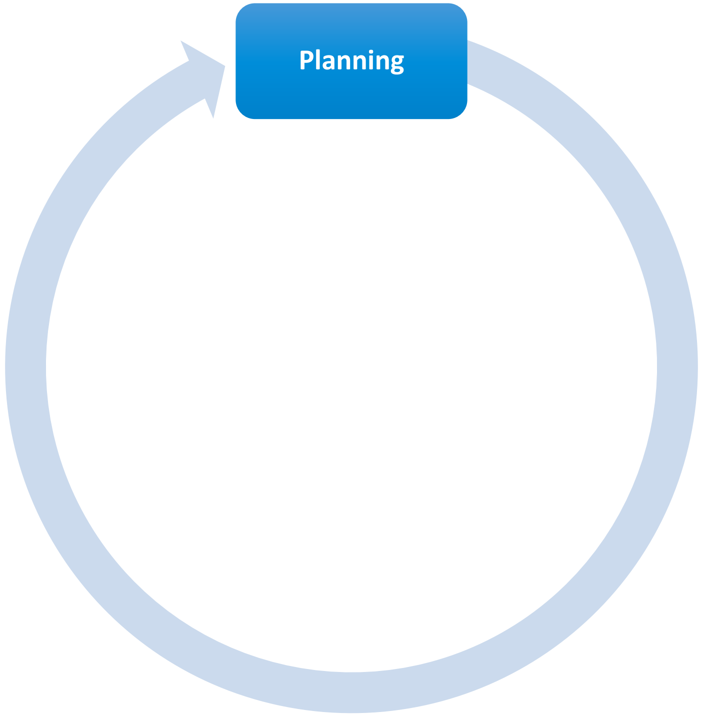


SDLC – WHAT IS IT?

Planning

Questions:

- When milestones of the project will happen?
- How much would it cost?
- Who would do what?
- How many people would we need?



Who does it:

- **Project Manager (PM)**
- Team Lead
- Team

What are major deliverables and milestones?

Analysis

Questions:

- How business is done now?
- What should change with new system?
- What system should DO? (functional requirements)
- How the system should BE? (non-functional requirements)

Planning
Analysis

Who does it:

- **Business Analyst (BA)**
- Team Lead
- Marketing
- User Experience (UX)
- Architect

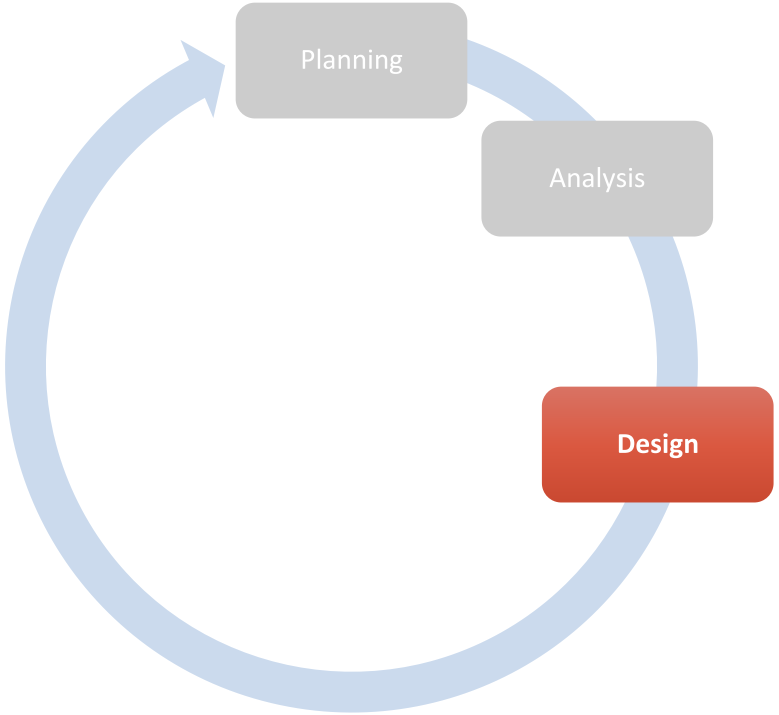
What system should BE and DO?

Design: architecture

Questions:

- How non-functional requirements will be fulfilled?
    - Hardware
    - Platforms
    - Technologies
    - Languages
    - Databases
    - Cloud providers
    - Deployment schemas
    - Architectural patterns

- How does the system fit into existing infrastructure?



Who does it:

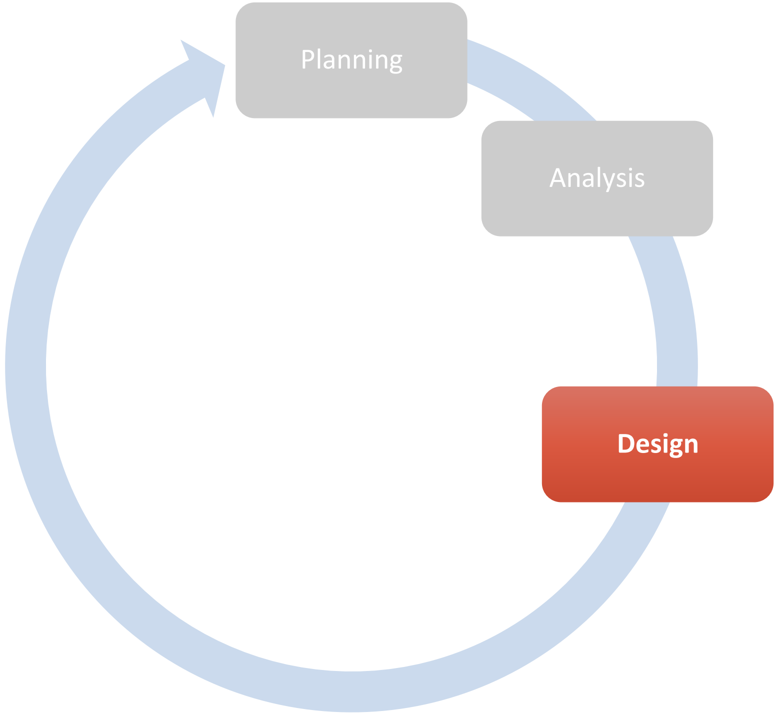
- **Architect**
- Team Lead

What is the target system infrastructure?

Design: components

Questions:

- How functional requirements will be fulfilled?
    - UI mockups, UX flow
    - Tests
    - Domain objects and interactions
    - Algorithms
    - Modules
- How to keep system maintainable, testable, portable, secure, scalable on code level?



Who does it:

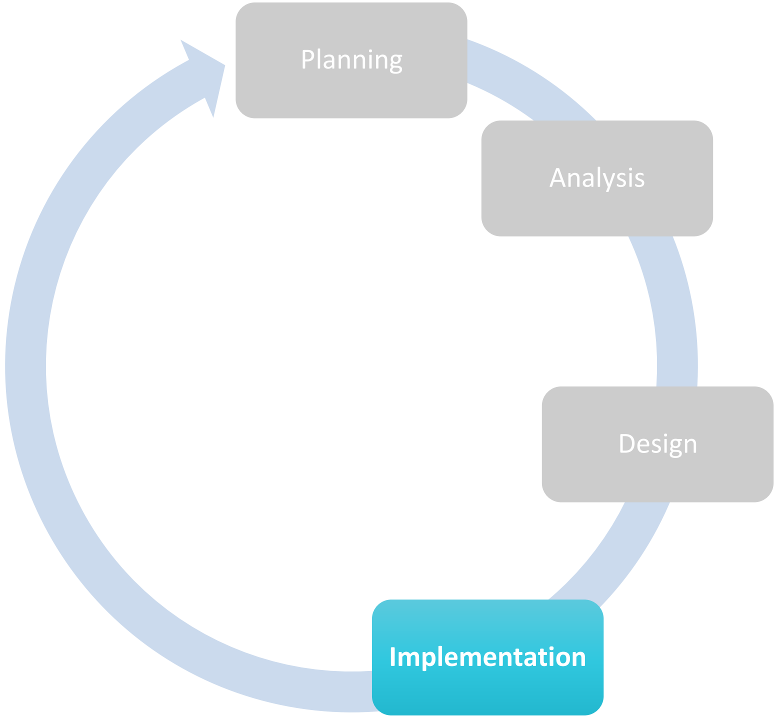
- **Team Lead**
- Developers
- Testers

How system functions are distributed in the infrastructure?

Implementation

Questions:

- How design should be translated into programming language?
- How do we make sure that code is doing the right thing?
- How do we keep code small, clean and readable?
- How multiple people would work with the same code?
- How do we keep track of the code of multiple application versions?
- What and how do we build?



Who does it:

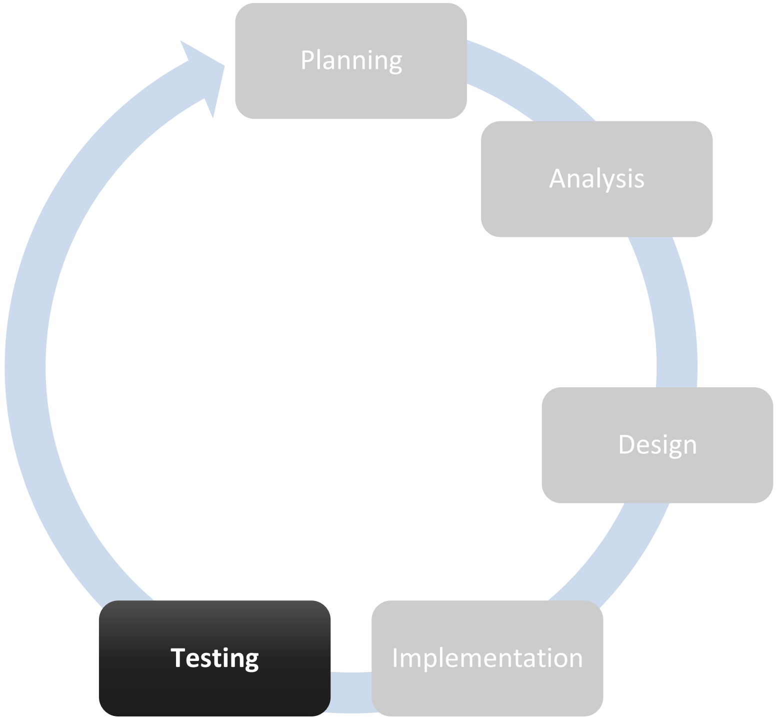
- **Developers**
- Team Lead
- Testers

How do we develop the software?

Testing and Integration

Questions:

- Do the system parts work and interact as expected?
- Does the system match the requirements?
- How to test the system in production-like environment safely?



Who does it:

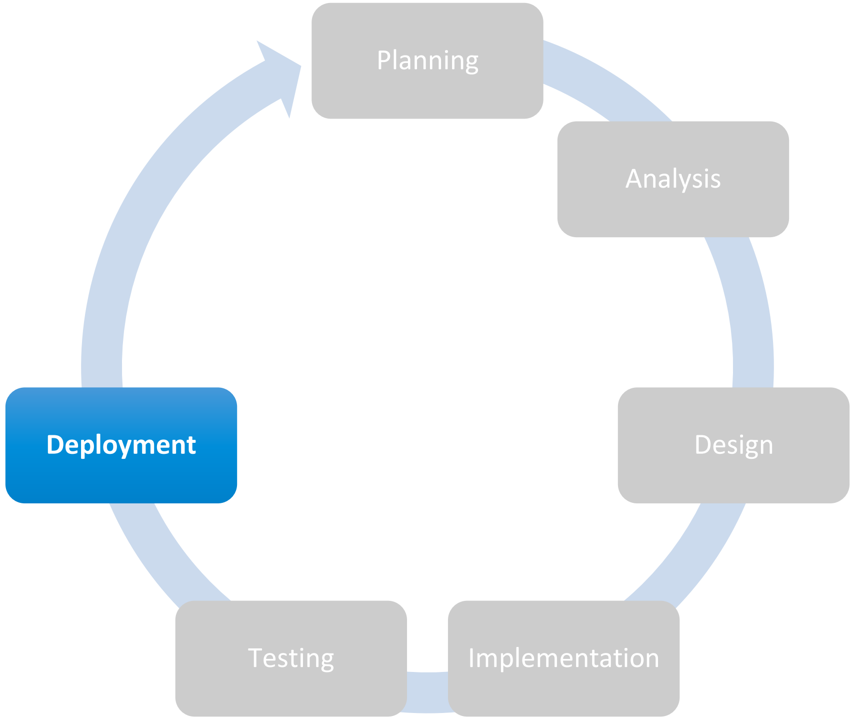
- **Testers**
- Automation Testers
- Developers

How is the system tested?

Deployment

Questions:

- How do we allocate infrastructure for the project?
- How the build reaches dedicated infrastructure?
- How configuration should be applied to dedicated infrastructure?
- How do we know if deployment made properly?



Who does it:

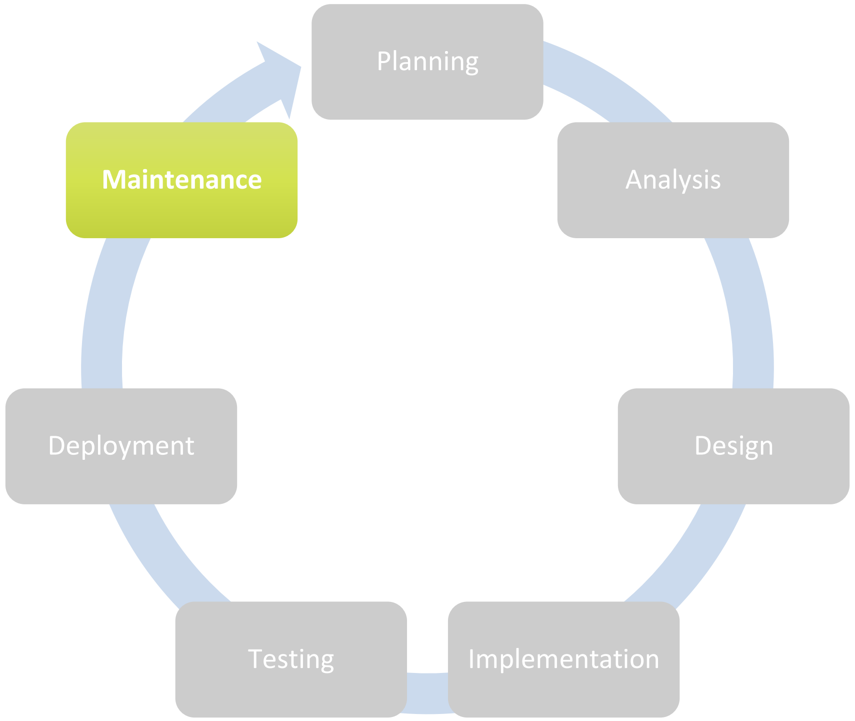
- **System Engineers**

How to deploy properly?

Maintenance

Questions:

- How do we keep the system running smoothly?
- How do we handle support requests from users?
- How do we investigate and handle incidents?
- How do we tune system, without stopping it?

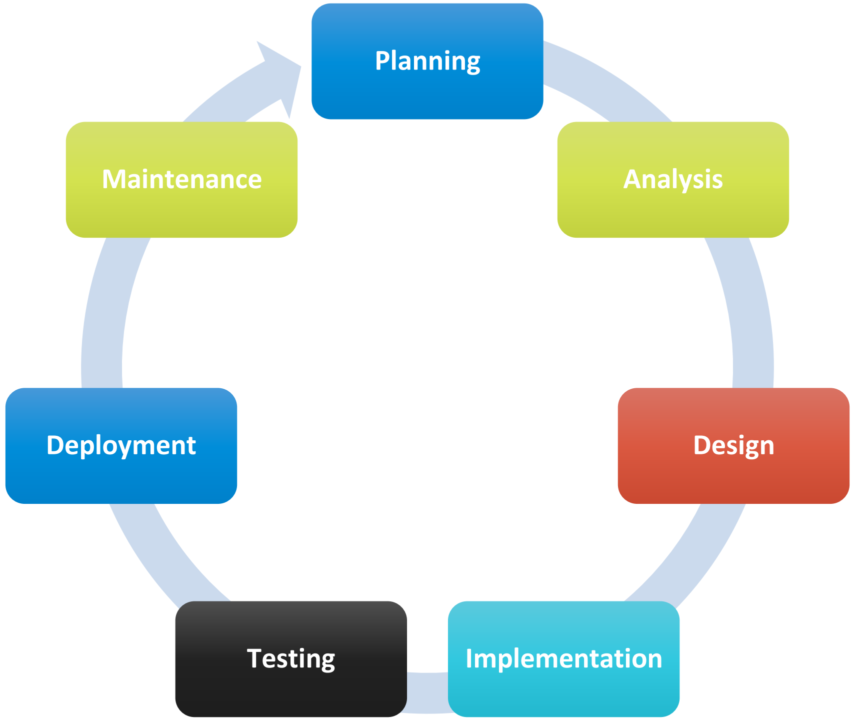


Who does it:

- **Application Support**
- System Engineers
- Developers

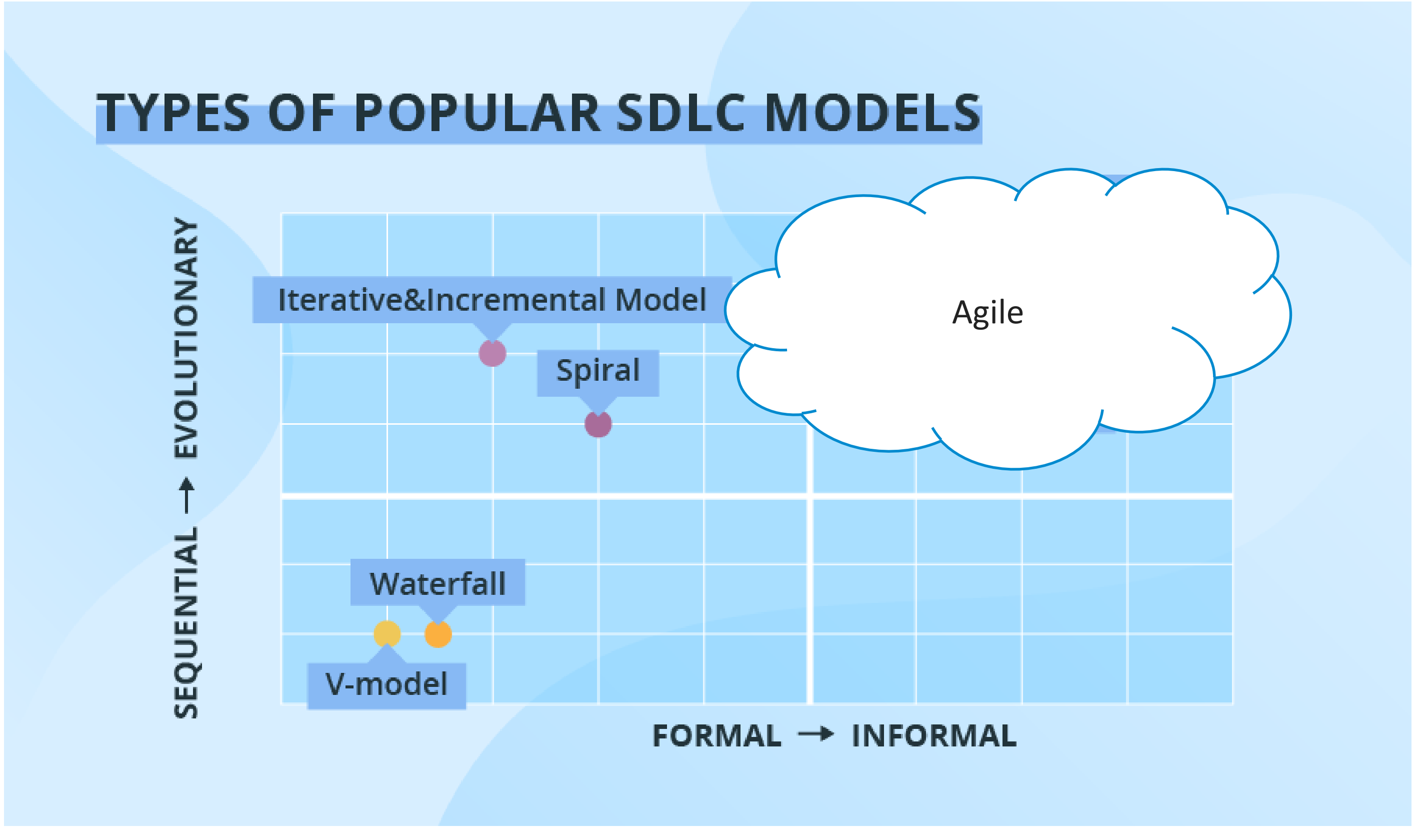
How to keep users happy with running system?

The Software Development Life Cycle

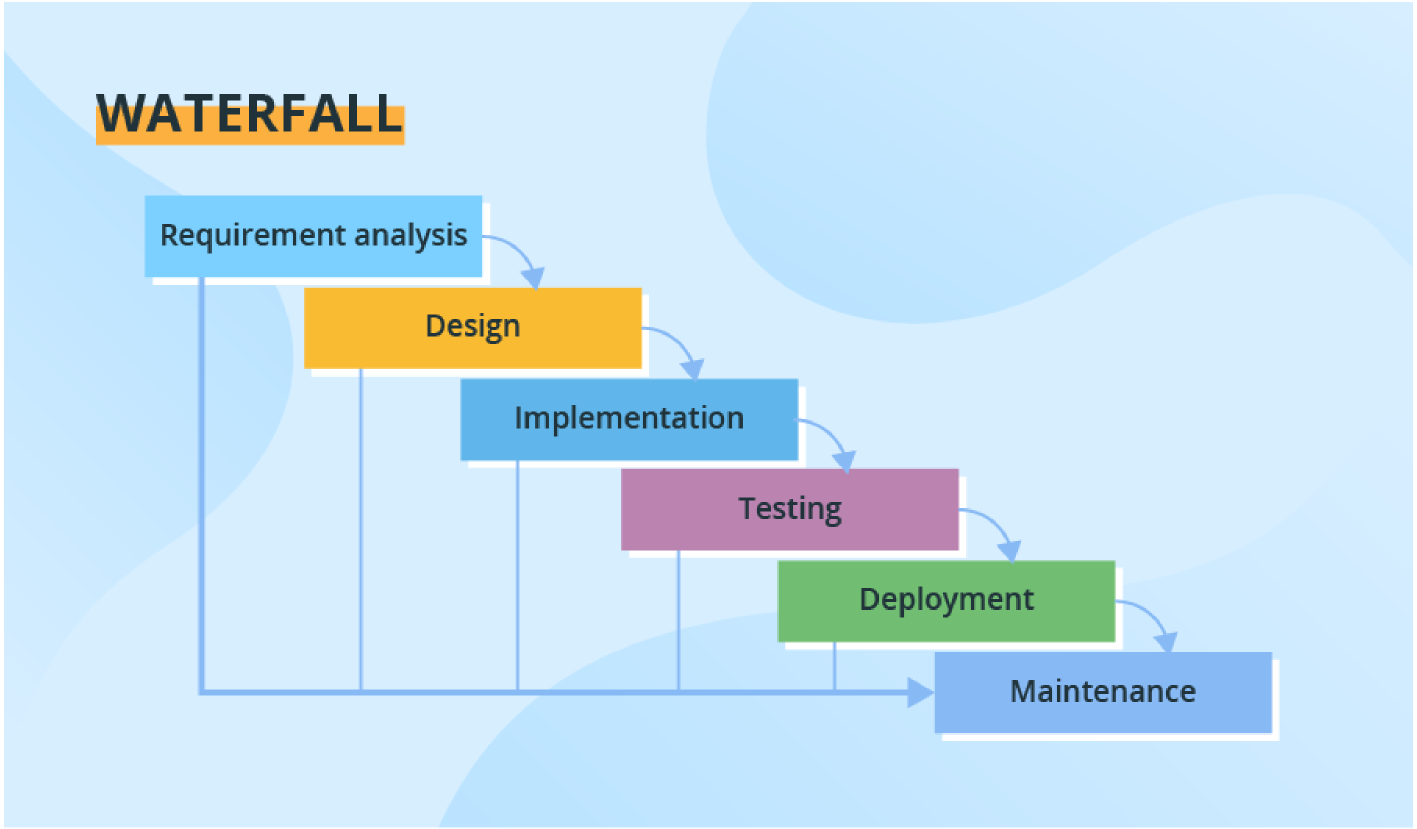


SOFTWARE DEVELOPMENT MODELS

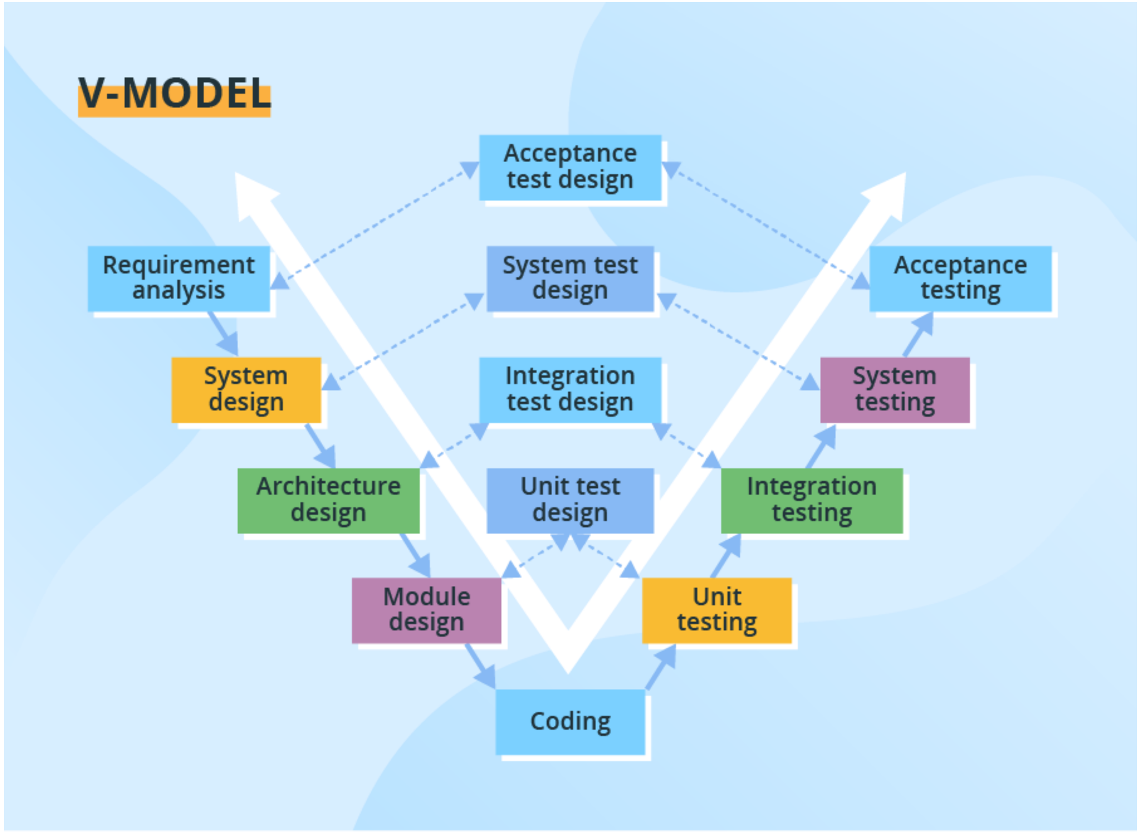
Types of SDLC models


TYPES OF POPULAR SDLC MODELS
Iterative&Incremental Model
Spiral
Agile
Waterfall
V-model
SEQUENTIAL → EVOLUTIONARY
FORMAL → INFORMAL

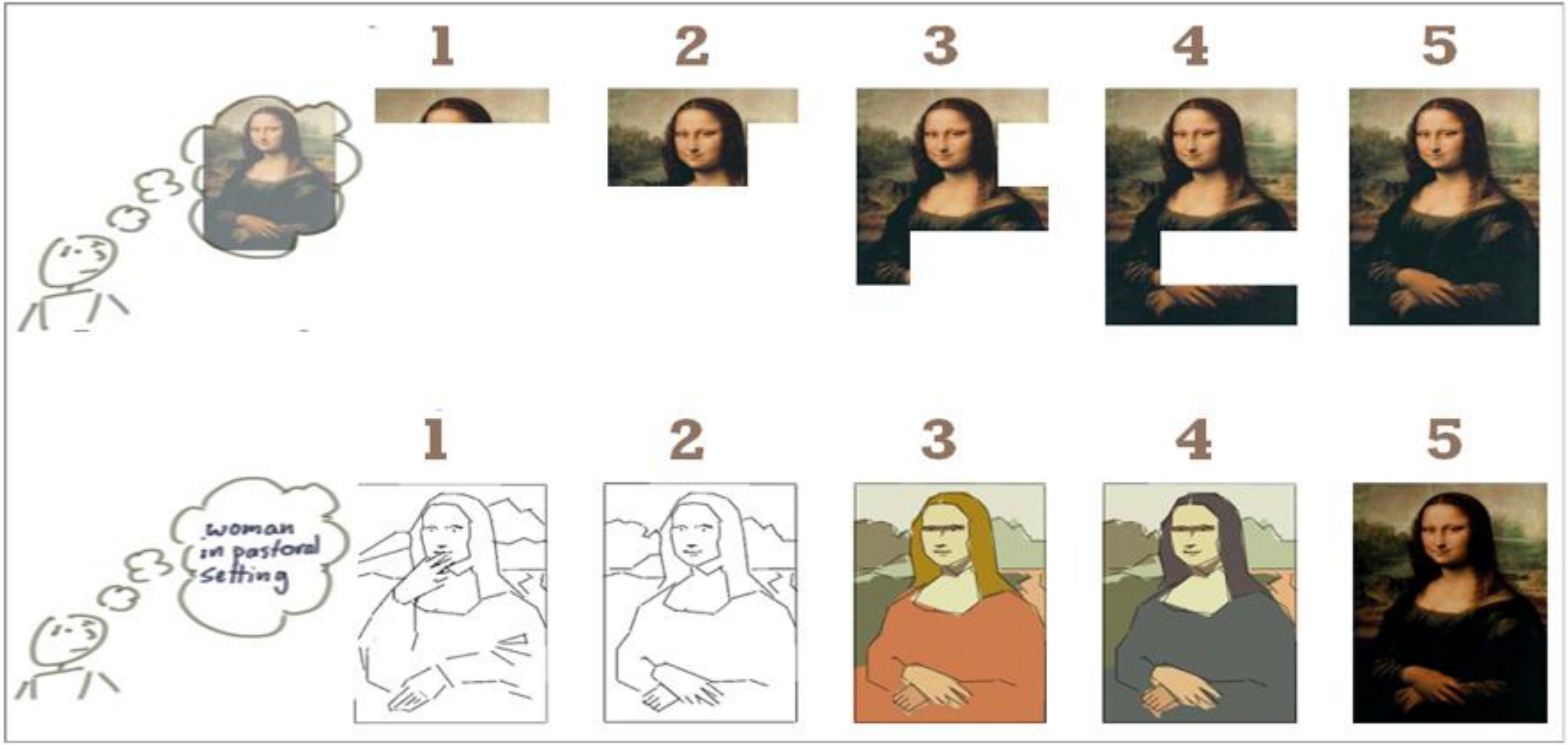
Sequential model



V-model (Validation and Verification model)



Incremental vs Iterative



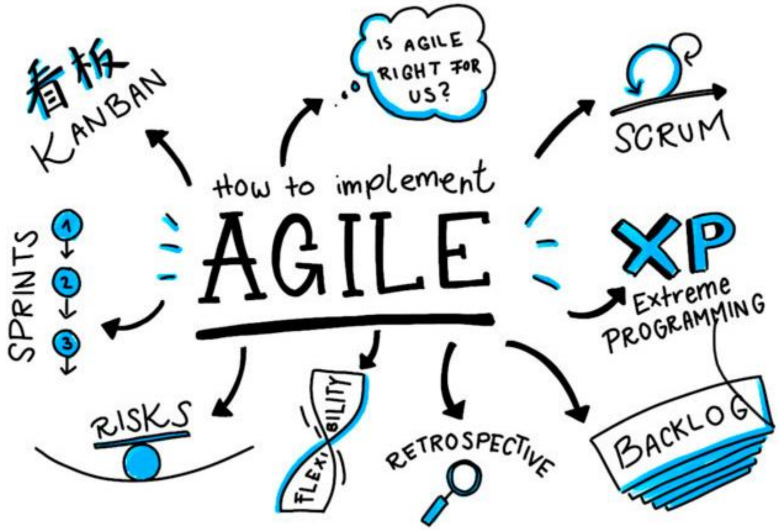
AGILE – A BIG FAMILY OF PRACTICES, METHODOLOGIES, FRAMEWORKS

What’s agile?

- Agile is an iterative approach to project management and software development that helps teams **deliver value** to their customers **faster** and with **fewer headaches**.

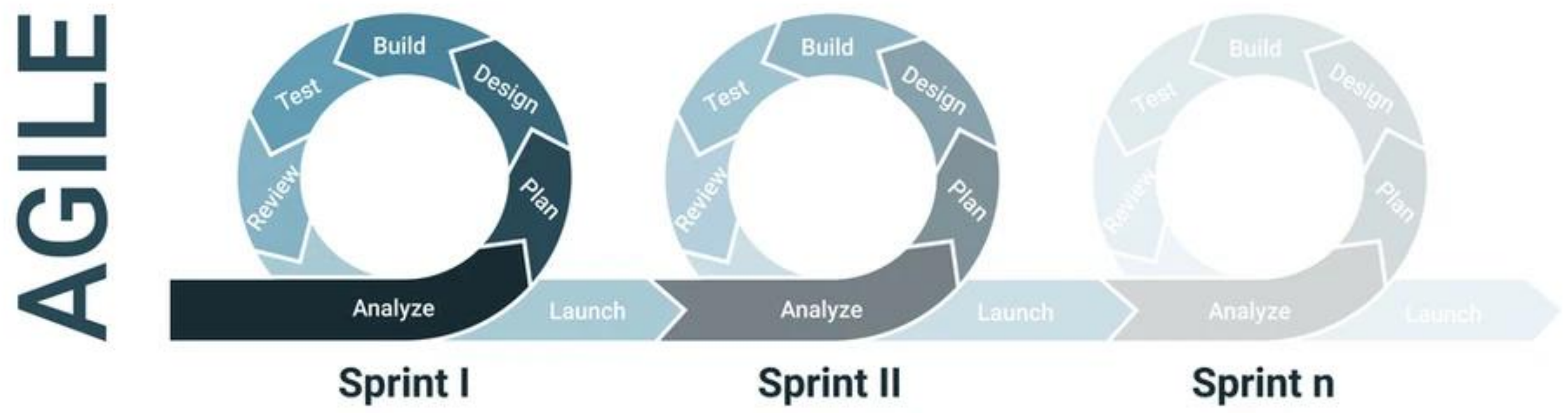
- Instead of betting everything on a "big bang" launch, an agile team delivers work in **small, but consumable, increments**.

- Requirements, plans, and results are evaluated **continuously** so teams have a natural mechanism for **responding to change quickly**.



Iteration & increment in Agile

WORKING INCREMENT IS THE GOAL OF EACH SPRINT



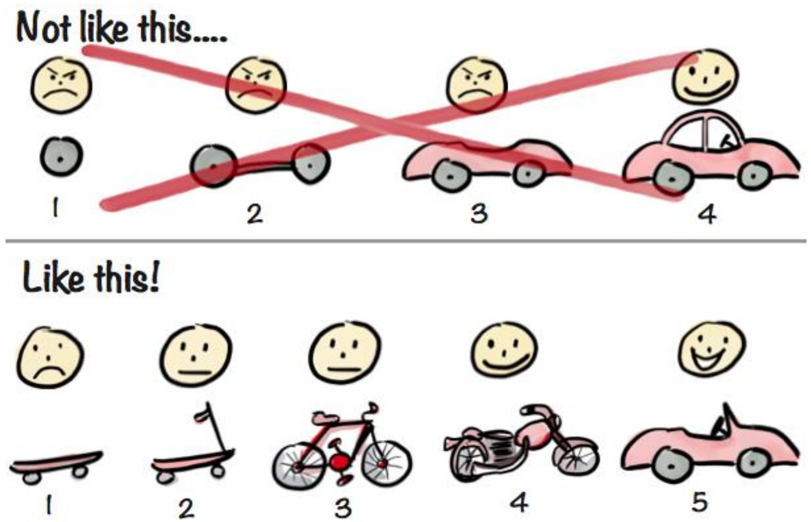
Minimum valuable product

MINIMUM VIABLE PRODUCT (MVP)

A particular version of the product that presents good features for the satisfaction of early customers to collect exclusive customer feedback manufactured for proper product development in the future.

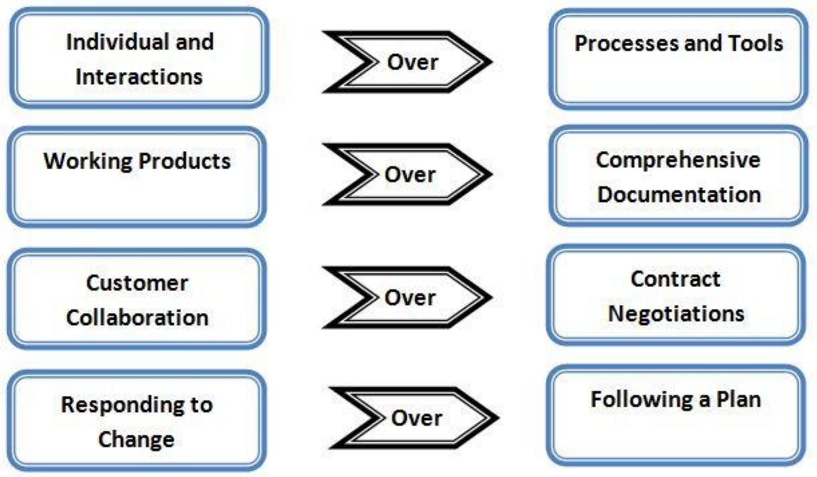
PURPOSES

•It helps in accelerating learning.
•The minimum viable product aids in reducing the wasted hours deployed in engineering the product.
•A minimum viable product (MVP) establishes a builder's ability to create products and justify its requirement.
•MVP envisages the quick building of a brand as any flaw in the product gets dealt with at an initial stage without reaching the masses. Hence improved versions of the product are supplied in the market, which further builds its goodwill and popularity.
•Minimal resources are required to improve the large production of goods, and supply is sent to early customers as soon as possible.



Agile manifesto and principals

MANIFESTO



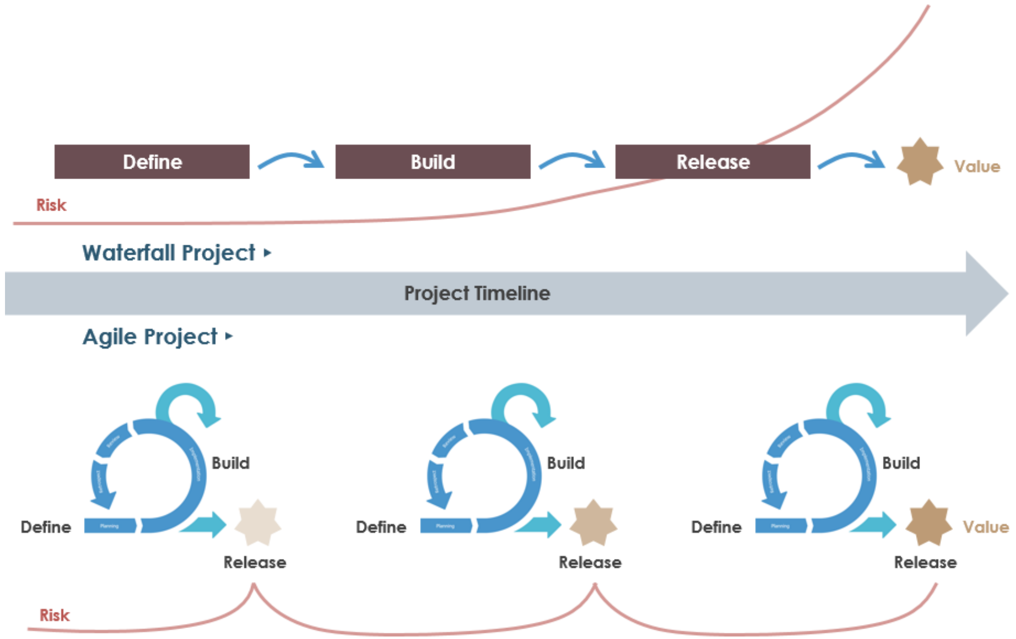
THAT IS, WHILE THERE IS VALUE IN THE ITEMS ON THE RIGHT, WE VALUE THE ITEMS ON THE LEFT MORE

PRINCIPLES

12 AGILE PRINCIPLES BEHIND THE AGILE MANIFESTO

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 a 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 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
7 Working software is the primary measure of progress.	8 The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	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.

Agile vs Waterfall



WATERFALL

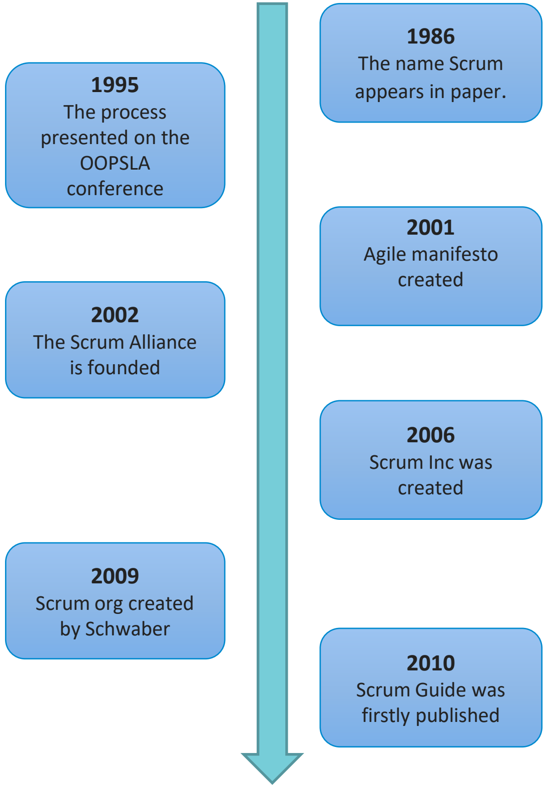
- Little is expected to change from the initial project scope
- Clear idea of the time required, budget, and resources
- It’s important that the final outcome is exactly what was planned and specified
- Rigid
- Highly unlikely that priorities will change
- The project is separated into distinct sections
- Low client involvement
- Little room for changes in time, budget, or resources

AGILE

- Changes can occur along the way
- Time required, budget and resources can change
- The final outcome can differ from the initial plan provided
- Flexible
- Priorities could change and the project needs to adapt
- The order of completion is irrelevant
- High client involvement
- Can adapt to changes

SCRUM – A LIGHTWEIGHT FRAMEWORK THAT HELPS TO GENERATE VALUE

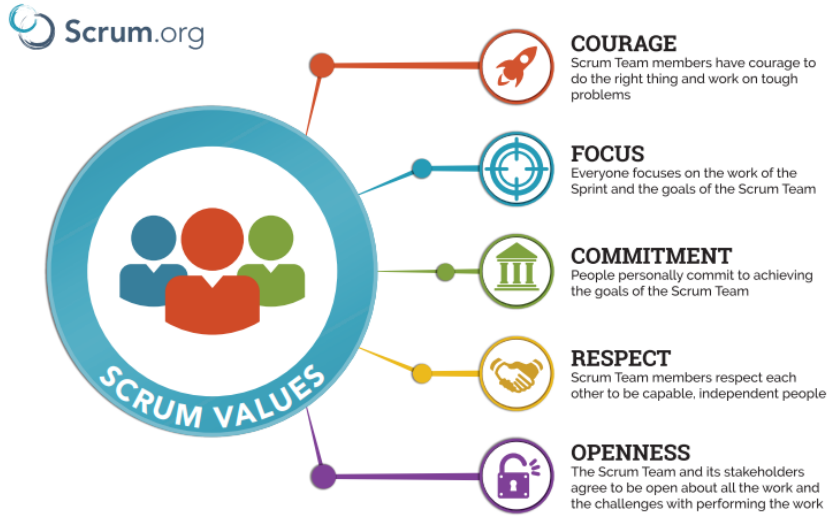
How was SCRUM developed


1986
The name Scrum appears in paper.
1995
The process presented on the OOPSLA conference
2001
Agile manifesto created
2002
The Scrum Alliance is founded
2006
Scrum Inc was created
2009
Scrum org created by Schwaber
2010
Scrum Guide was firstly published

- The SCRUM was developed by Jeff Sutherland and Ken Schwaber

- The first Scrum Guide was published in 2010
- Scrum is now over 26+ years old (young!) and its use continues to grow. It is now widely used outside of software development.

SCRUM values


Scrum.org
SCRUM VALUES
COURAGE
Scrum Team members have courage to do the right thing and work on tough problems
FOCUS
Everyone focuses on the work of the Sprint and the goals of the Scrum Team
COMMITMENT
People personally commit to achieving the goals of the Scrum Team
RESPECT
Scrum Team members respect each other to be capable, independent people
OPENNESS
The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work

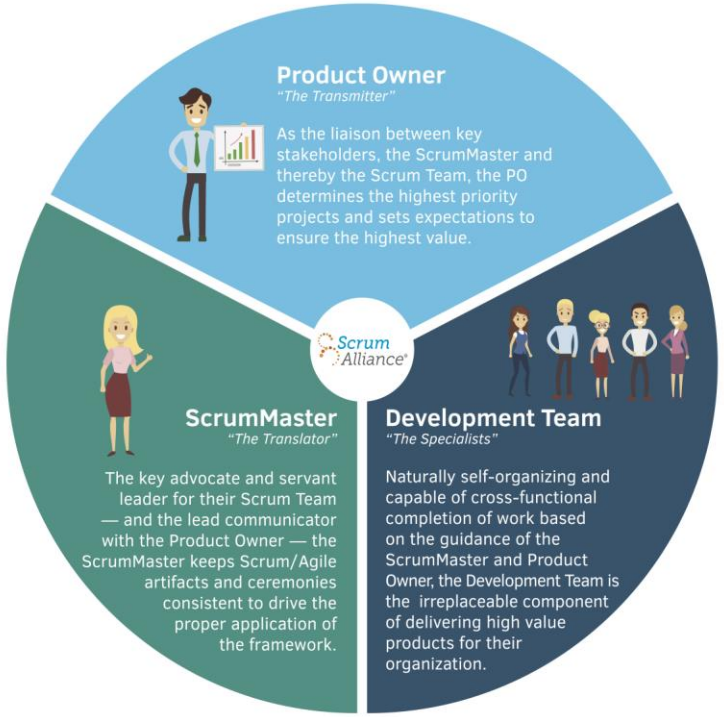
The truth is that values are more than words or abstract ideas. If they don't lead to concrete changes in behavior, you may as well not have them.

SCRUM Roles

SCRUM MASTER

DEV TEAM

PRODUCT OWNER



Product Owner
"The Transmitter"

As the liaison between key stakeholders, the ScrumMaster and thereby the Scrum Team, the PO determines the highest priority projects and sets expectations to ensure the highest value.

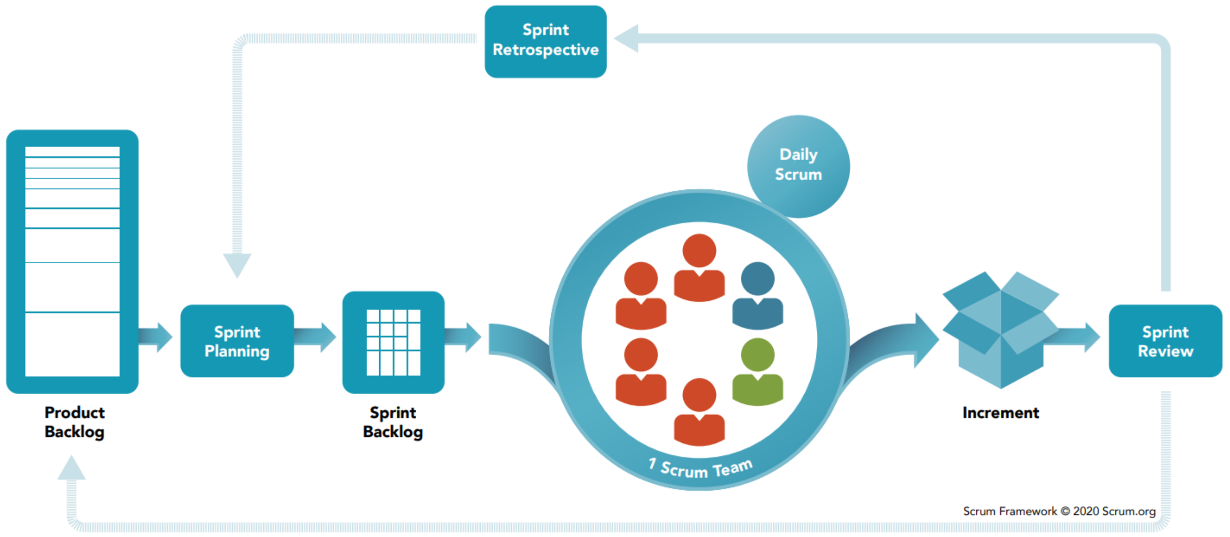
ScrumMaster
"The Translator"

The key advocate and servant leader for their Scrum Team — and the lead communicator with the Product Owner — the ScrumMaster keeps Scrum/Agile artifacts and ceremonies consistent to drive the proper application of the framework.

Development Team
"The Specialists"

Naturally self-organizing and capable of cross-functional completion of work based on the guidance of the ScrumMaster and Product Owner, the Development Team is the irreplaceable component of delivering high value products for their organization.

Events



THE EVENTS ARE MANDATORY

- ❑ Sprint planning
- ❑ Sprint
- ❑ Daily scrum
- ❑ Sprint review
- ❑ Sprint retrospective

Scrum metrics

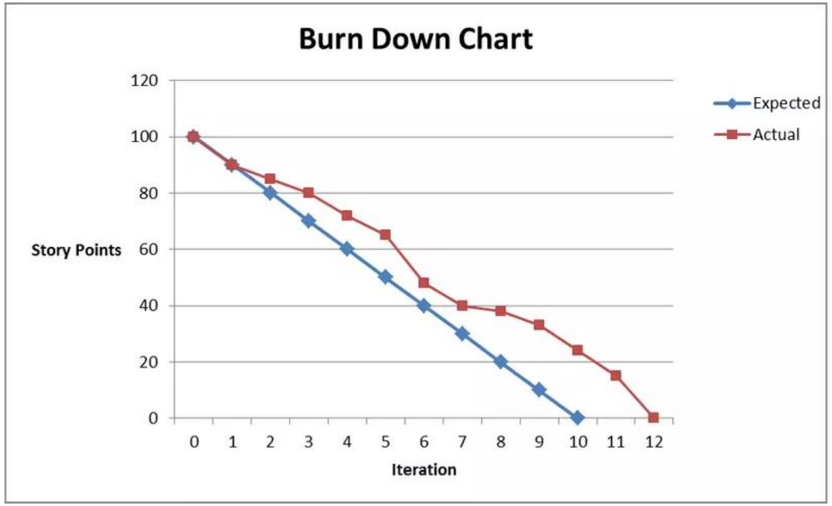
WHAT ARE SCRUM METRICS?

Scrum metrics are specific data points that scrum teams track and use to improve efficiency and effectiveness. When defined, understood, and implemented, scrum metrics can become insights that help guide and improve a team's agile journey.

WHY DO WE NEED SCRUM METRICS?

Scrum metrics can help teams establish benchmarks and guide the direction of the work. For this reason, scrum metrics are helpful for established and new teams alike. Tracking scrum metrics also helps bring visibility to various dimensions of a team's effectiveness, whether it's a team's velocity, capacity, predictability in delivery, or quality of the product. Key metrics can cultivate awareness in the team's performance and instigates action to change and improve. Plus, they can even help to gauge team happiness and satisfaction over time.

ONE EXAMPLE – BURN DOWN CHART



KANBAN – FROM JAPAN WITH LOVE

History

THE KANBAN IS THE METHOD BORN FROM LEAN METHODOLOGY

Toyota invented this approach in the middle of the twentieth century as a way to streamline its production of cars and eliminate wasted time and resources. (Any action that did not impact the functionality of the car being built and delivered was considered a waste under this system, and therefore removed from the process.)

Eventually, other manufacturing organizations across many industries began using this system, and the name later changed to Lean. The methodology was first applied to the creation of software in 2003 with the publication of the now-famous book Lean Software Development.

Principles & practices

PRINCIPLES

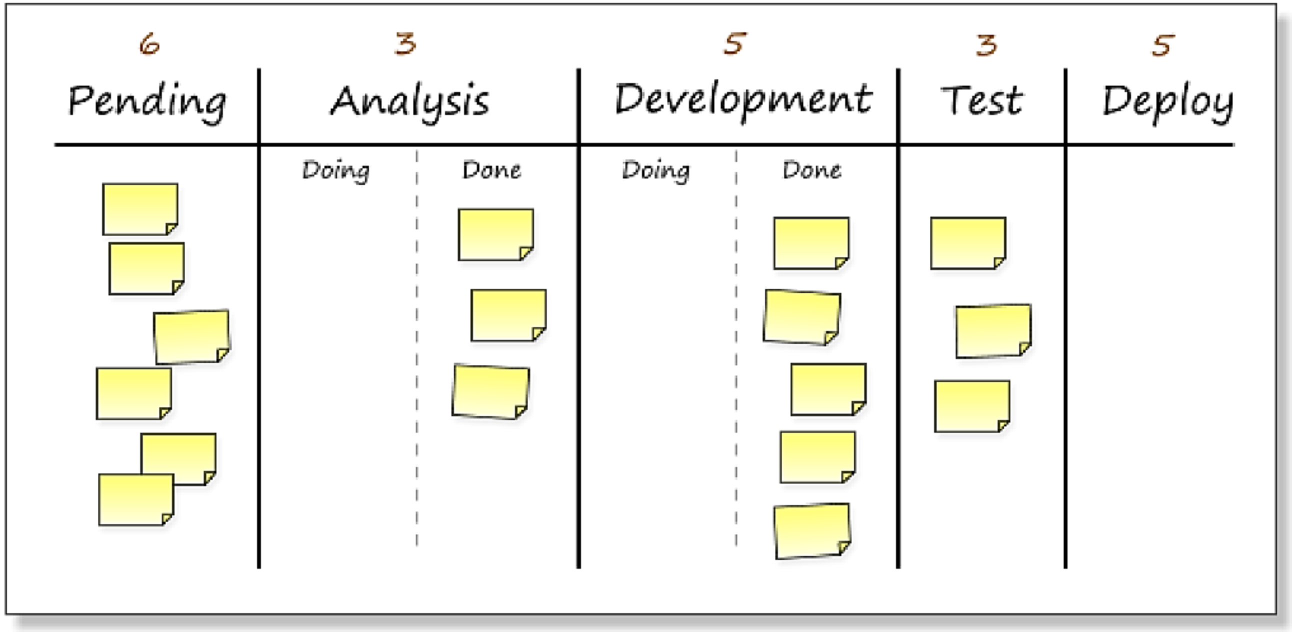
- Start with what you do now
- Agree to pursue incremental, evolutionary change
- Respect the current processes, roles, responsibilities & titles
- Encourage acts of leadership at all levels

PRACTICES

- Visualize the workflow
- Pull instead of Push
- Limit work in progress
- Manage flow
- Make process policies explicit
- Implement feedback loops
- Improve collaboratively & evolve experimentally

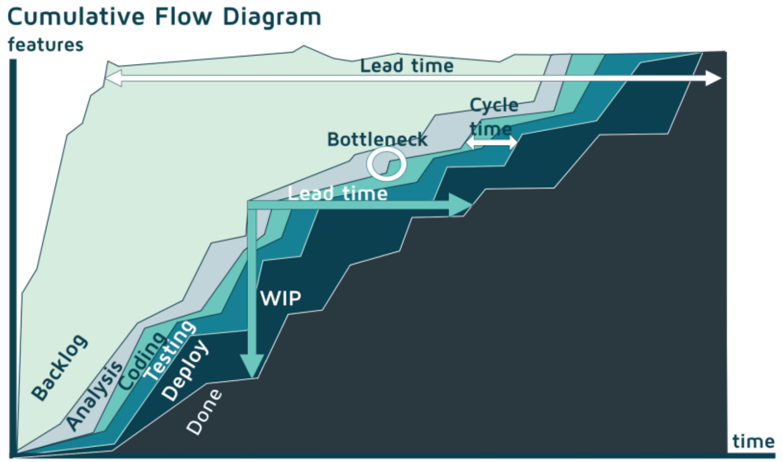
Boards

ALMOST THE IDEAL BOARD



Kanban metrics

CUMULATIVE FLOW DIAGRAM



METRICS

- **Lead time** measures how much time a task spends from creating to be fully done
- **Cycle time** is a key metric in Kanban and it measures how much time a task spends going through your process. Cycle time differs from lead time in that it is only measured from when your team starts working on the task.
- **WIP** which is measured along the vertical axis, directly impacts the time to value – Cycle time and Lead time. Reducing WIP in conjunction with balancing WIP limits for a smooth flow provides for the most effective business value realization.
- **Bottleneck**. Increasing thickness of any individual band indicates a bottleneck in that stage of the workflow. This could be being created due to delays in that step or because of a hold up in the next step in the process. This should trigger a discussion within the team on how to resolve and to remove the bottleneck.

DEVOPS ROLE IN SDLC

The wall of confusion

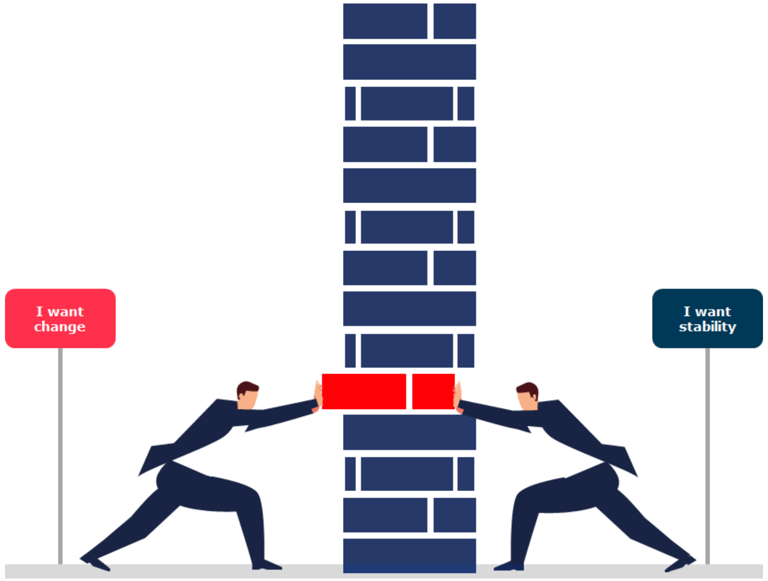
DEVOPS IS NO LONGER AN OPTION — EITHER YOU ADOPT IT OR YOU LOSE

- It was a constant struggle between the dev and ops teams:

Dev: threw software packages over the wall to the operations team

Ops: how to convert the software into deployable, production-ready material?

This wall of confusion extends beyond the mindsets of the two teams with the tools they use. The dev team used some tools, and the operations team used something else to perform the same stuff.

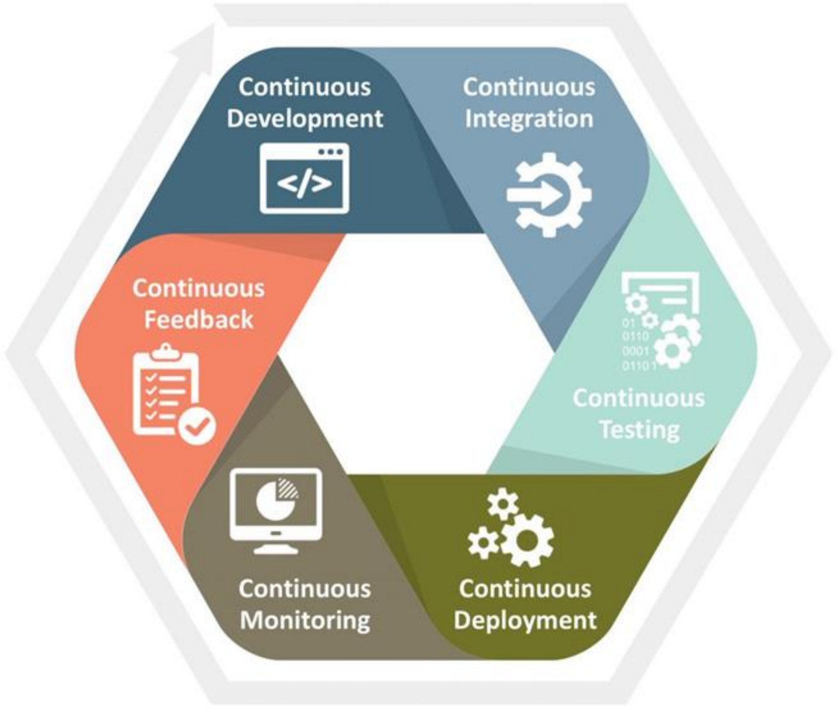


Uniting Dev and Ops

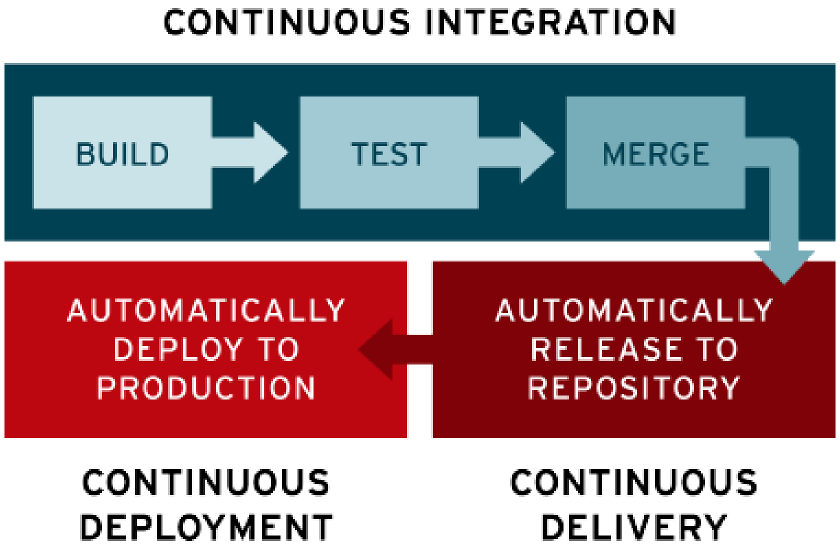
DevOps demolishes the walls between the development and operations team, unifying development to operations for better, faster outcomes by automating the software delivery lifecycle using a set of processes and tools.

Technology concepts of Devops

- Version control
- Automated testing
- Test-driven development
- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Configuration control
- Cloud calculation
- Infrastructure automation
- Containers
- Monitoring, logging, visualization



CI/CD/CD



- **Continuous integration -**

the practice where developers merge the changes to the code base to the main branch as often as possible.

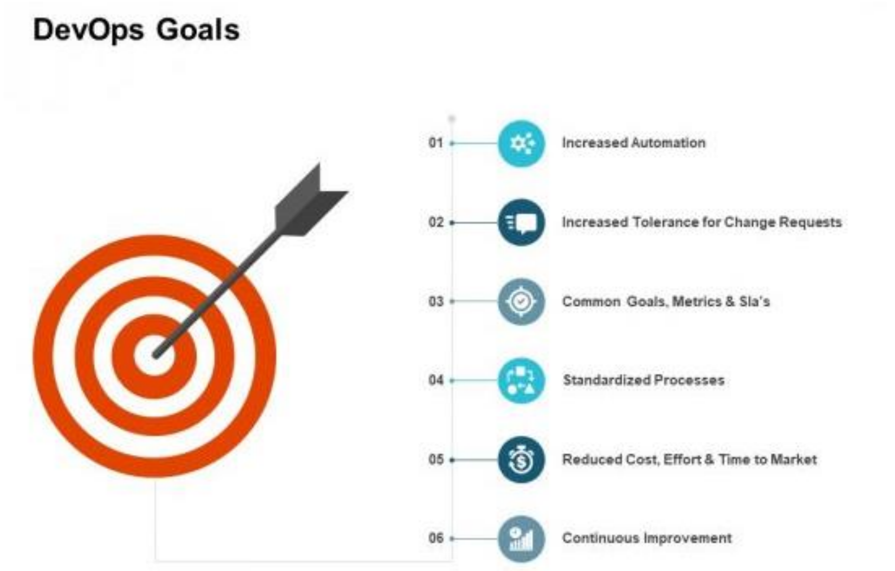
- **Continuous delivery -**

is an extension of CI since it enables automation to deploy all the code changes to an environment (dev, qa, stage, prod, etc) after the changes have been merged.

- **Continuous deployment -**

takes the process one step further than continuous delivery. Here, all changes that pass the verification steps at each stage in the pipeline are released to production.

DevOps goals


DevOps Goals
01 Increased Automation
02 Increased Tolerance for Change Requests
03 Common Goals, Metrics & Sla's
04 Standardized Processes
05 Reduced Cost, Effort & Time to Market
06 Continuous Improvement

DevOps Benefits:

- Faster, better product delivery
- Faster issue resolution and reduced complexity
- Greater scalability and availability
- More stable operating environments
- Better resource utilization
- Greater automation
- Greater visibility into system outcomes
- Greater innovation

THANK YOU!