Software Process Models

Waterfall, V-Model, Iterative, Incremental, Spiral, Agile, DevOps Toolchain, CI/CD Pipeline

Software Process

- A software process model is an abstract representation of the development process.
- Software process models identify what development activities in what order (sequence) should be done.
- The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.
- Software process models are also referred as SDLC models.

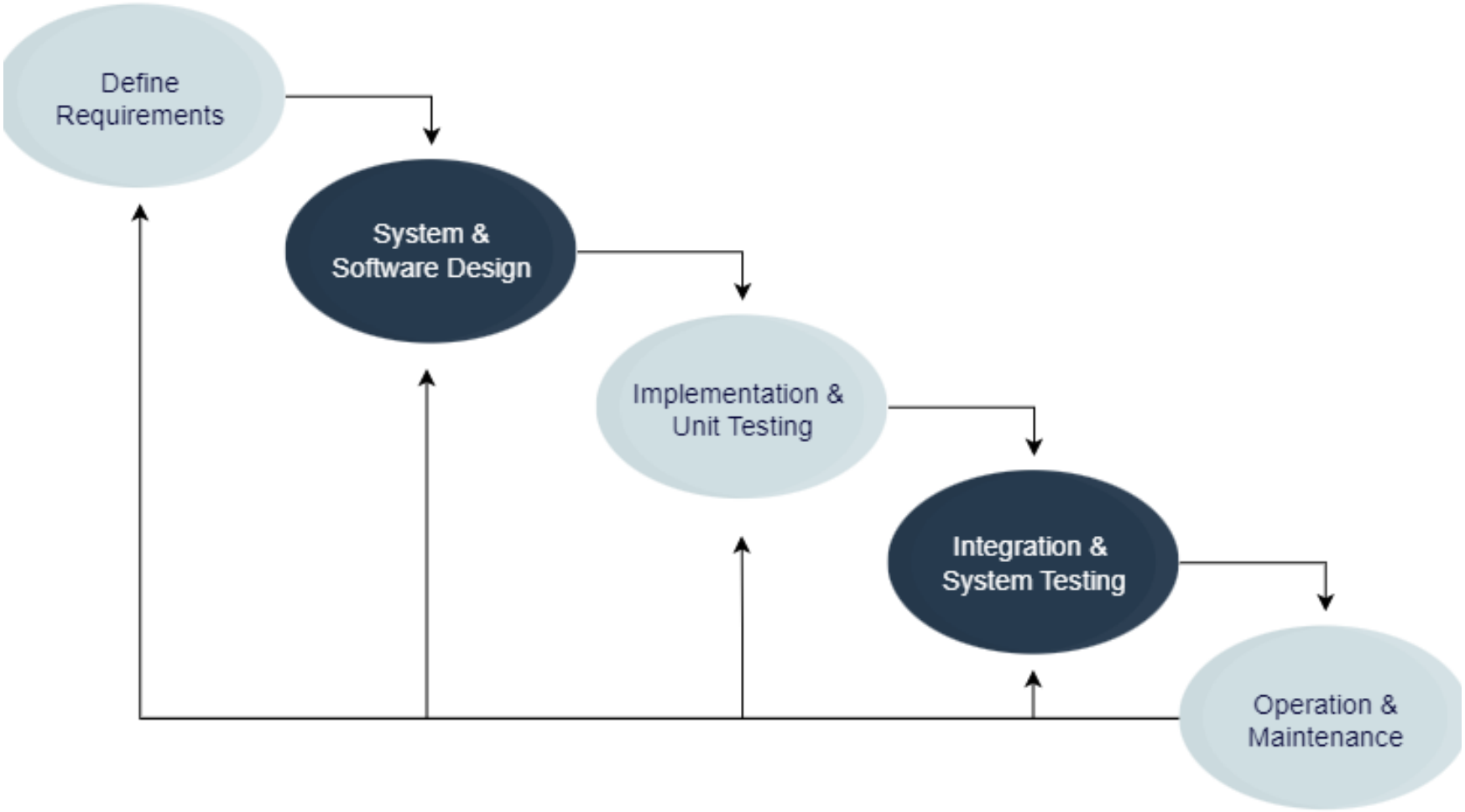
Popular SDLC Models

- Waterfall model
- V model
- Iterative model
- Prototype model
- Spiral model
- Incremental model
- Agile Model
  - Scrum
  - Kanban
  - XP

Waterfall Model

- The waterfall model is a sequential, plan-driven process where you must plan and schedule all your activities before starting the project.
- Development Phases:
  - Requirements Analysis
    - In this phase, a large document called Software Requirement Specification (SRS) is created which contained a detailed description of what the system will do in the common language.
  - Software Design
    - All this work is documented as a Software Design Document (SDD)
  - Implementation
  - Testing
  - Deployment
  - Maintenance

Waterfall Model (Cont'd)



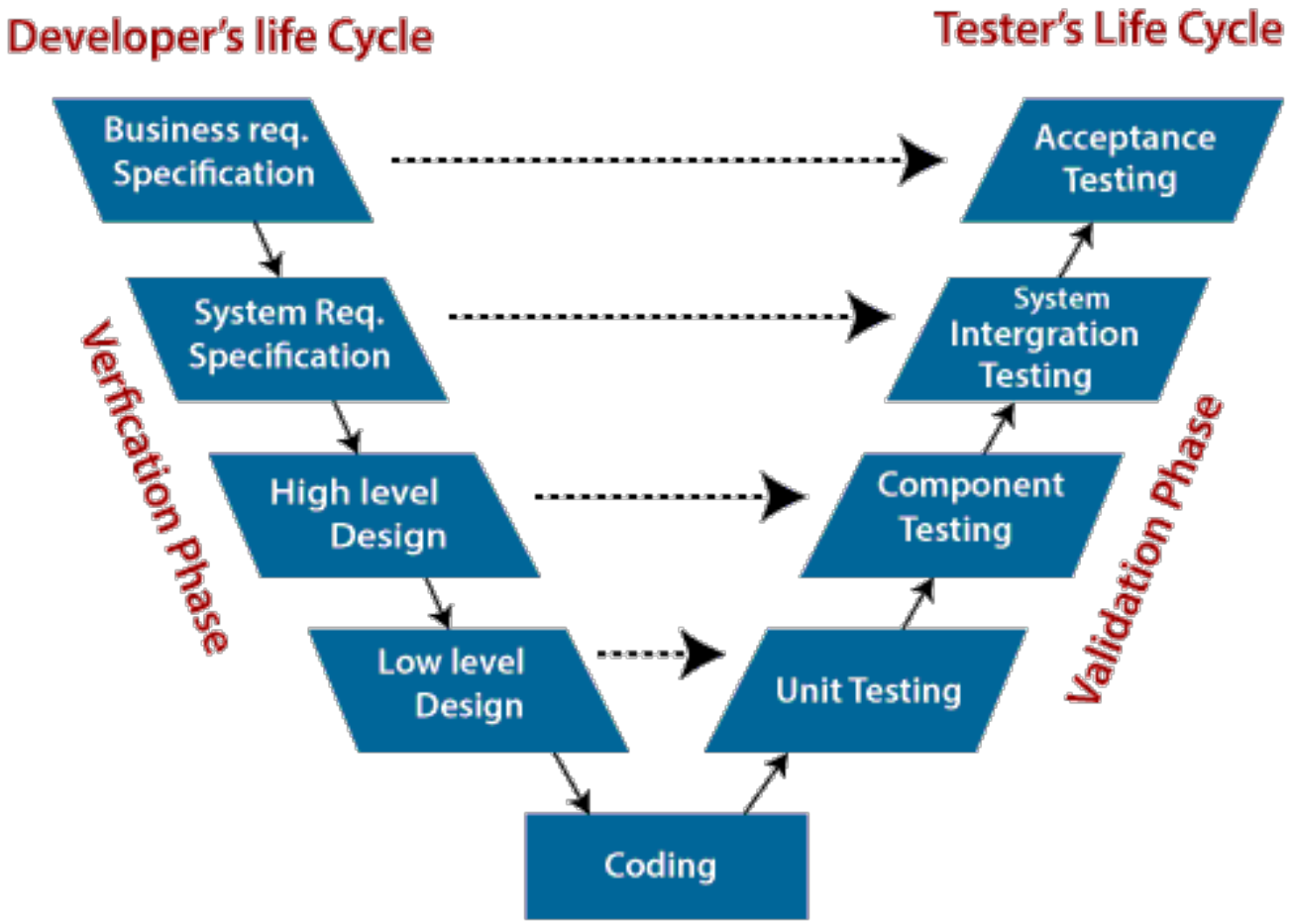
Waterfall Model: Pros and Cons

- The waterfall model is easy to understand and follow.
- It doesn't require a lot of customer involvement after the specification is done.
- Since it's inflexible, it can't adapt to changes.
- There is no way to see or try the software until the last phase.
- The waterfall model has a rigid structure, so it should be used in cases where the requirements are understood completely and unlikely to radically change.

V Model

- The V model (Verification and Validation model) is an extension of the waterfall model.
  - Software Verification: Are we building the product right?
  - Software Validation: Are we building the right product?
- All the requirements are gathered at the start and cannot be changed.
- You have a corresponding testing activity for each stage. For every phase in the development cycle, there is an associated testing phase.

V Model (Cont'd)



V Model: Pros and Cons

- The V model is highly disciplined, easy to understand, and makes project management easier.
- The V model isn't good for complex projects or projects that have unclear or changing requirements.
- The V model is a good choice for software where downtimes and failures are unacceptable; i.e. systems with high reliability requirements.

Iterative Model

- Iterative means that a team delivers work frequently rather than all at once.
- In an iterative model, you will also first create a rough overall plan of all necessary work steps. But in contrast to the classical "waterfall", this overall plan is then divided into small steps, the iterations. The next iteration is planned in detail and then implemented.
- Iterative models include iterations that are actually refinements applied on the previous iteration.
  - The first version of software has some basic implementation of all the requirements. The next version improves the previous version, …

Iterative Model (Cont'd)

https://www.jpattonassociates.com/dont_know_what_i_want/

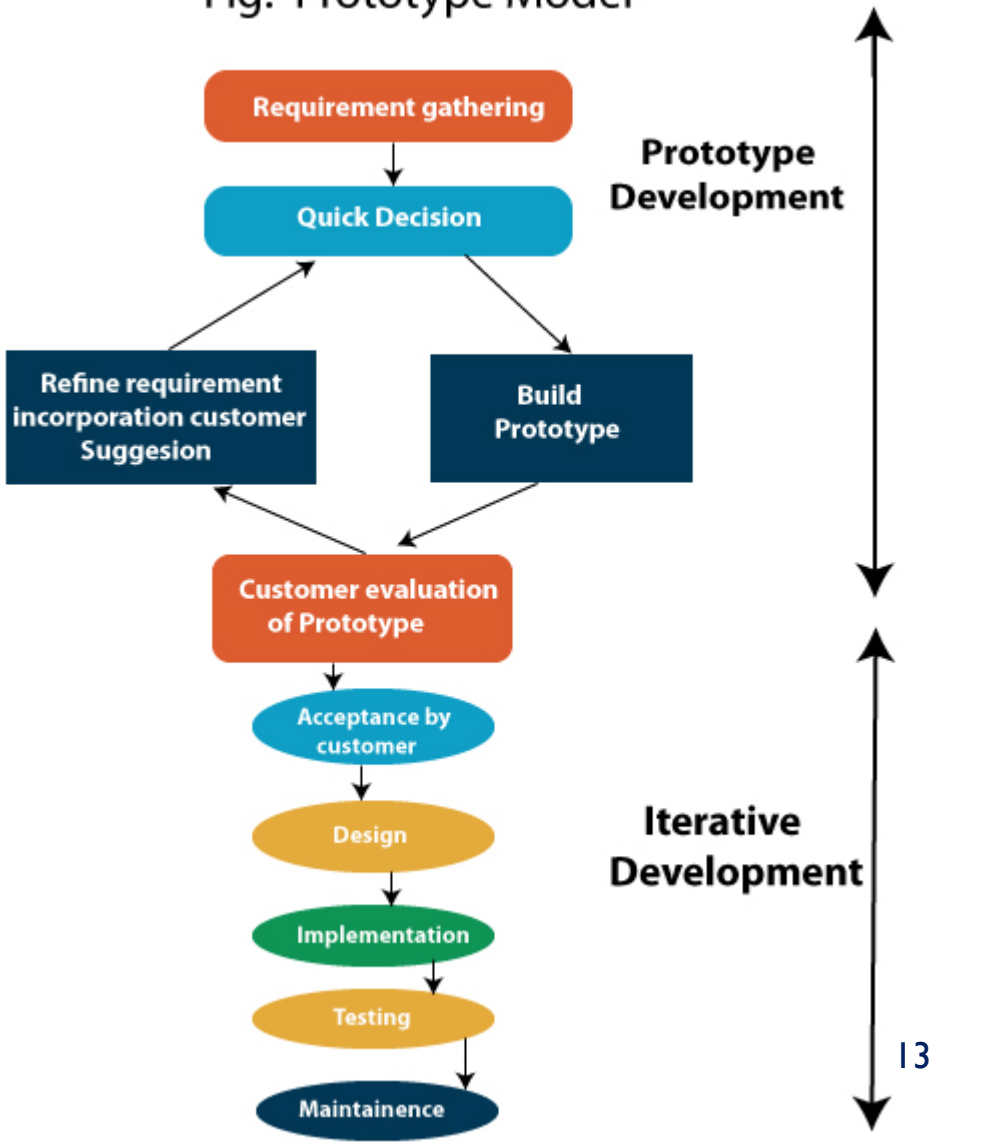
Iterative Model: Pros and Cons

- A team working iteratively thus has the possibility to incorporate the knowledge gained from a completed iteration into the planning of the next iteration.
- Still, the overall plan is relatively rigid, for example, with regard to the scope of features. The flexibility results primarily from the details how to implement the features in detail:
  - The more high-level the requirements are, the more flexibility is gained.
- Project management and risk management is more complex than in a traditional(waterfall) development.

Prototype Model

- In prototype model before developing the actual software, a working prototype of the system is built.
- A prototype is a simulation of the system with limited functionality, low reliability, and low efficiency.
- Prototypes are categorized in:
  - Low fidelity prototypes
  - High fidelity prototypes

Fig: Prototype Model

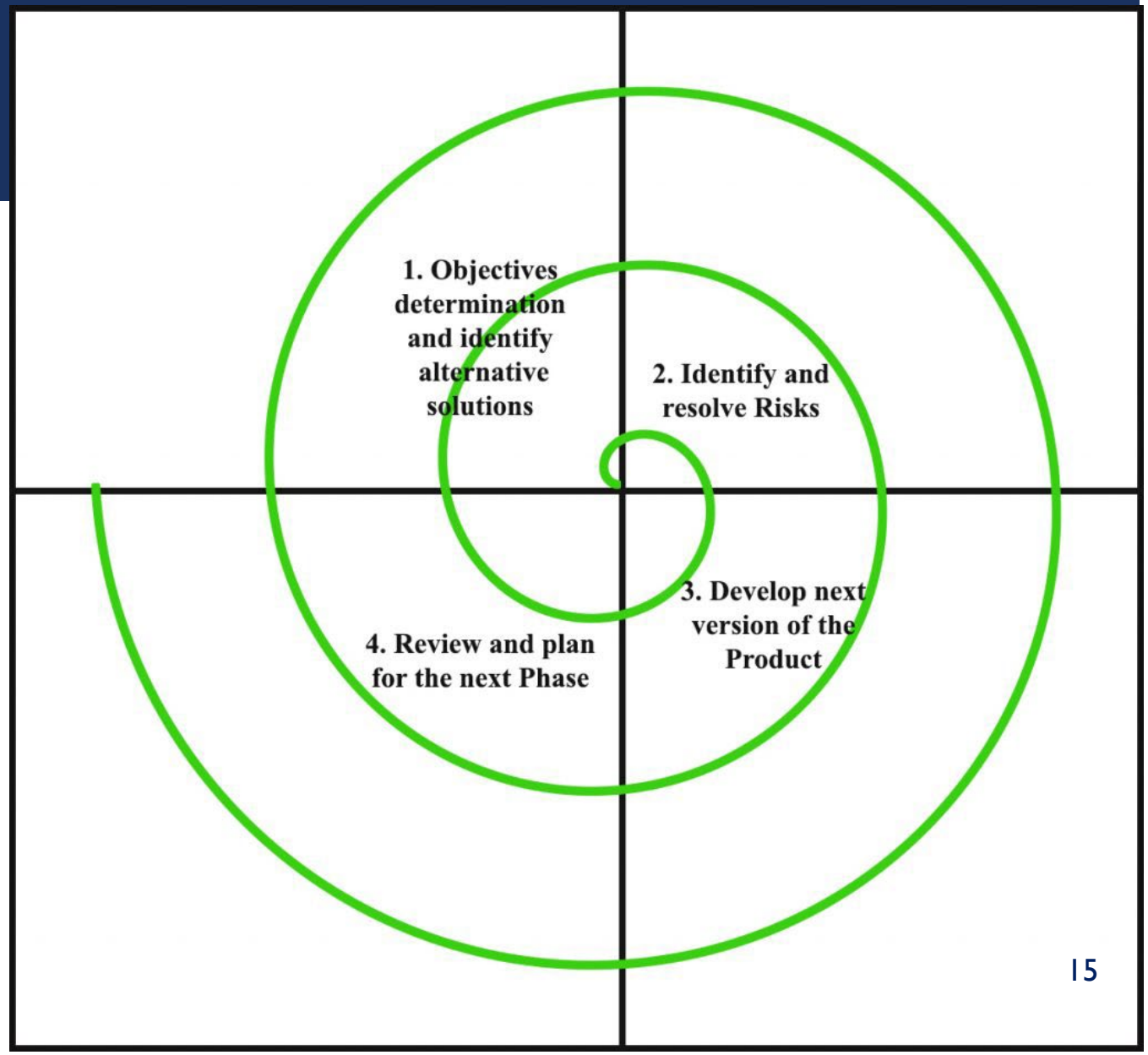


Prototype Model: Pros and Cons

- Prototyping includes rework and is costly.
- Prototyping can be used as a testing bed with cheap failing.
- Prototyping can be used to clarify the system's requirements in the cases that the requirements are not clear.

The Spiral Model

- Spiral model is a risk-driven process which is a combination of waterfall, iterative, and prototype models.
- Each iteration (spiral) includes four phases and include all the steps of a waterfall model.
- In the second quadrant of the spiral alternative solutions are evaluated, risks are identified, and prototypes for the selected solution are built before starting the development phase.



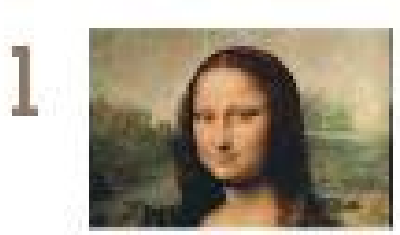
Spiral Model: Pros and Cons

- Spiral Model is very suitable for large and complex projects with medium to high risk.
- Change of the requirements can be incorporated in the next spiral.
- Project estimation in spiral model is difficult because the number of spirals are not specified.
- This model is very dependent on risk analysis and requires more experienced teams.

Incremental Model

- Incremental development is a process that focuses on small steps to reach the final product goal.
- The subsequent increments expand on the previous ones until everything has been updated and implemented.
- At the end of each increment a subset of the system is delivered as a complete working software.
- Each increment is built on the previous increments.

Incremental Model (Cont'd)



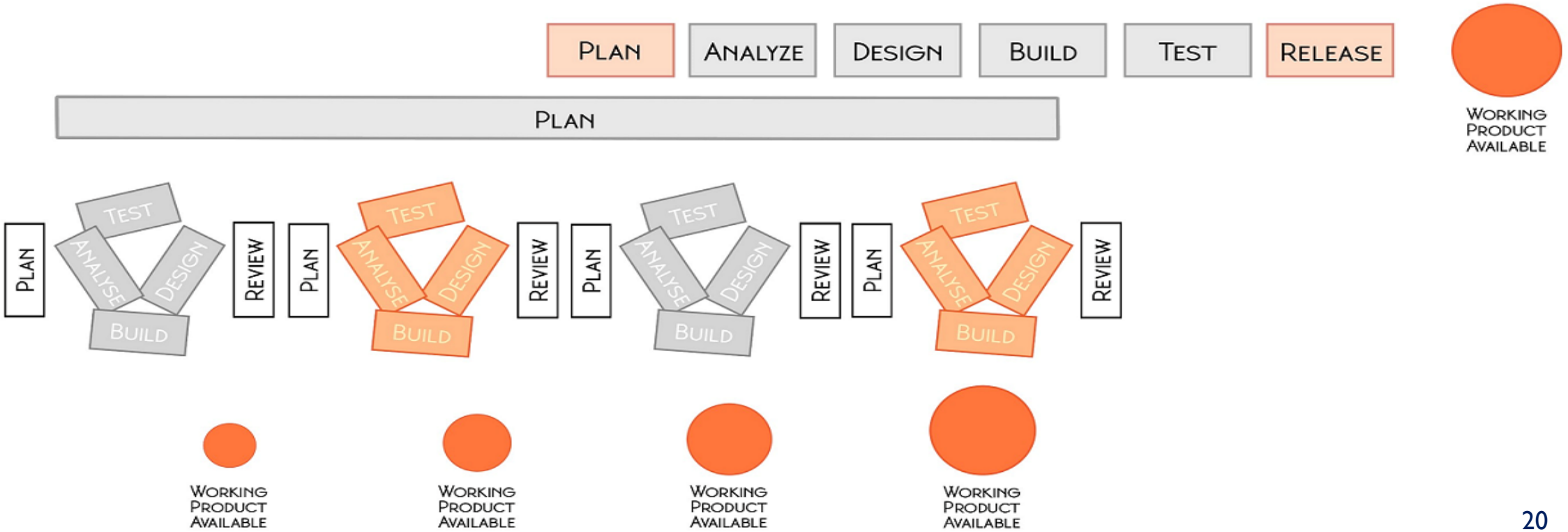
https://www.jpattonassociates.com/dont_know_what_i_want/

Incremental Models: Pros and Cons

- Parallel development is possible in incremental model.
- Early release is possible in incremental models.
- Incremental models are suitable for the cases that the requirements are known upfront.
- Change of scope and requirements are not too expensive.
- This model is more difficult to plan and design.

Agile Model

- Agile models are iterative-incremental models.



Agile Model (Cont'd)

- Development starts with the knowledge of major requirements that are implemented in the initial increments.
- In an agile process, planning is incremental and continual as software is developed.
- Agile Manifesto is a document that outlines the 4 core values and 12 principles of agile development model:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

Agile Model: Pros and Cons

- Because this method works with steady and consistent progression, it can adapt to changes that occur within the process, such as unexpected technological limitations or the implementation of new features.
- Working software is released earlier and faster.
- Since customer feedback is received after each increment/iteration, and is considered to plan next increment/iteration, the risk is more controlled in agile models.
- Agile development requires more experienced teams.
- Not collecting all the requirements at the beginning may produce some problems with the system's architecture.
- Agile models are highly flexible and suitable for large projects.

Agile Development Frameworks

- There are some well-known agile frameworks that define specific approaches to planning, managing, and executing work. Some of the most popular ones are:
  - scrum
  - Kanban
  - Extreme Programming (XP)

Scrum

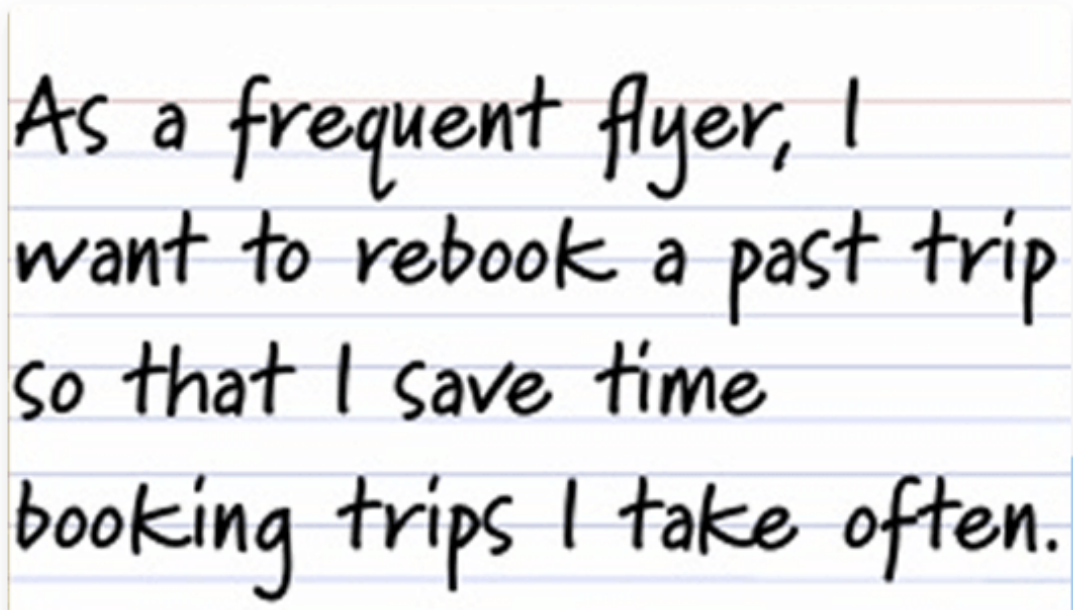
- Scrum is a very popular agile framework in which increments/iterations called sprints that are time-boxed.
- Scrum framework includes the following components:
  - Sprints
  - User stories
  - Product backlog
  - Release backlog
  - Sprint Backlog
  - Product owner
  - Scrum master
  - Burndown charts
  - Meetings: Daily Scrum and Sprint Retrospective

Scrum Components: User Stories

- User stories are features of the intended system written in a simple language from the viewpoint of an end user.
- User stories are written on index cards that are pinned to a visual board.
- In agile development anyone in the team can write the user stories, and user stories are written throughout the whole project.
- A story-writing workshop is usually held near the start of the agile project.

Scrum Components: User Stories (Cont'd)

Title:	Priority:	Estimate:
User Story: As a [description of user], I want [functionality] so that [benefit].		
Acceptance Criteria: Given [how things begin] When [action taken] Then [outcome of taking action]		



https://www.justinmind.com/

Scrum Components: Product Backlog, Product Owner, and Scrum Master

- The collection of user stories is called the product backlog.
- One of the roles of any scrum team is the product owner who is responsible to validate the product backlog to make sure that the features are aligned with the users' and customers' need.
- Another key role in scrum is the scrum master who sets up meeting and monitors the work being done. He also makes sure that all the tools that are required in the team are available, and the release is planned.

Scrum Components: Release Backlog

- In Scrum release planning starts from the product backlog. The team selects some of the user stories that they want to be included in the next release and add them to the release backlog.
- User stories in the release backlog are prioritized and stamped with their estimated time, and if required the user stories are broken into smaller tasks.
- The time estimate of the entire release then will be the summation of all the estimated tasks included.
- Stories are estimated in two ways:
  - Story points that identify the difficulty of that story (e.g. 1, 2, 4, …)
  - Hour, day, or month buckets (2h, 4h, 8h, etc.)

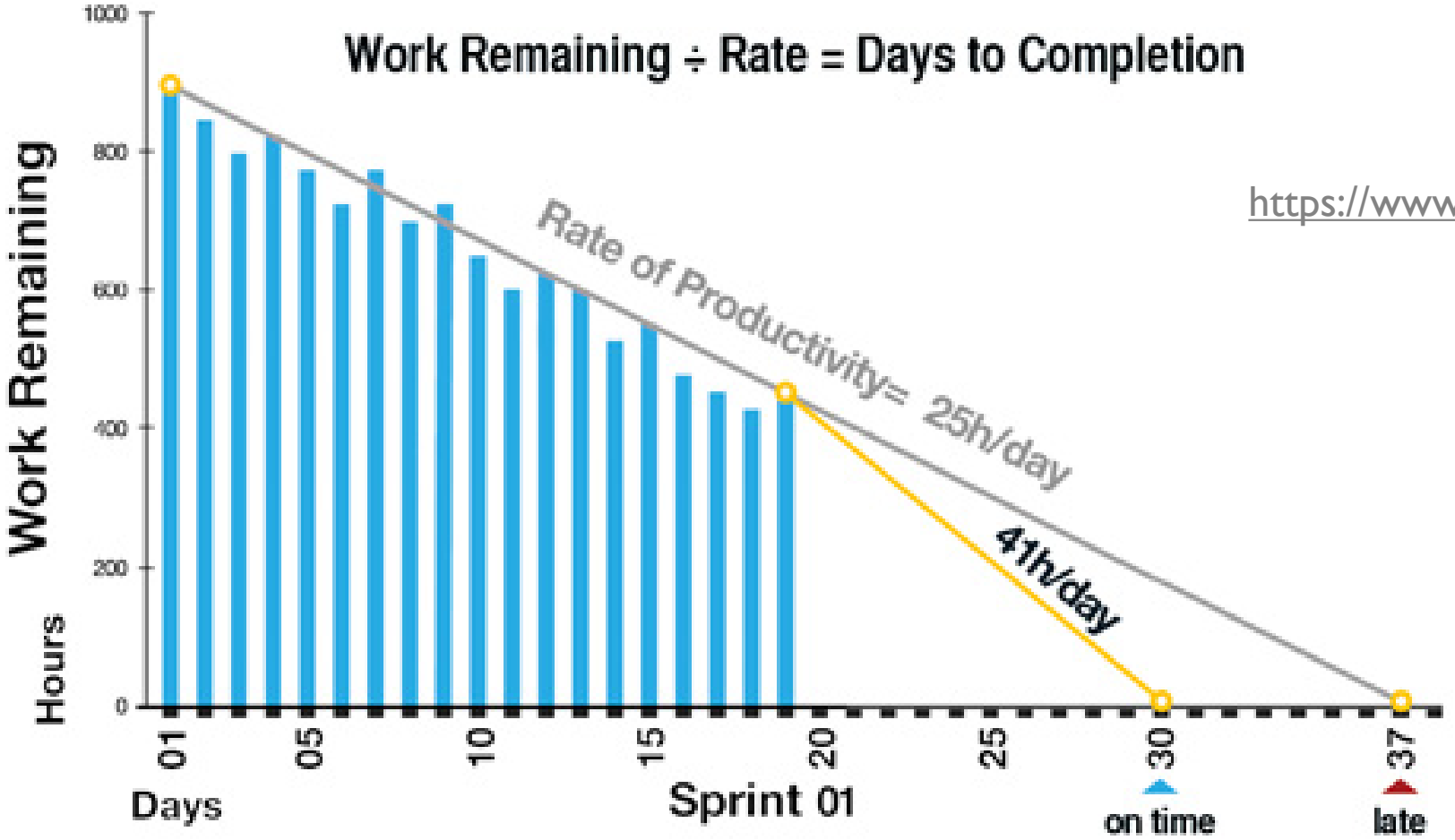
Scrum Components: Sprint and Sprint Backlog

- A sprint is a short, time-boxed period during which the team gets some tasks ready to ship. Each sprint is usually 1 to 4 weeks long depending on the release cycle that they belong to: shorter release cycles lead to shorter sprints.
- Each release has between 2 to 12 sprints.
- At the end of each sprint all the tasks included in the sprint backlog must be implemented and fully tested to be in the ship-ready state.
- Sprints are planned by sprint backlogs that include some tasks from the release backlog that are selected for that specific sprint.

Scrum Components: Burndown Chart

- Since each sprint includes a realistic estimate of a working part of the project, a late sprint shows that the project is not on the schedule and something needs to be done.
- A burndown chart is used to monitor sprints. Burndown charts show the amount of work remained in a sprint on a day-by-day basis.
- Data required for burndown charts are collected from developers on a daily basis.
  - Each developer identifies the time spent every day.
- Burndown charts are very effective visual tools that show the team velocity, and any possible delay in the sprint.

Scrum Components: Burndown Chart (Cont'd)



https://www.scrumhub.com/

Scrum Components: Daily Scrum and Retrospective Meeting

- Daily Scrums are short, fast-paced, daily, standup meetings in which every team member shortly communicates with the other team members what has been done since the last meeting and if any obstacles exists.
- Daily meetings ensure that any major issues are addressed as soon as possible.
- At the end of each sprint a retrospective meeting happens in which team members discuss what went right and what are the areas of improvement.

Kanban

- Kanban is an agile development framework that visualizes the work using a Kanban board, Kanban cards, and WIP limit.
- By visualizing the work, Kanban ensures that every team member understands the work better and everyone is on the same page.
- A Kanban board is organized into columns that from left to right show the workflow of the team: the stages that the tasks need to go through. The workflow could be less or more complex in different teams so the Kanban board may have less or more columns accordingly.
- Kanban is a pull system.

Kanban (Cont'd)

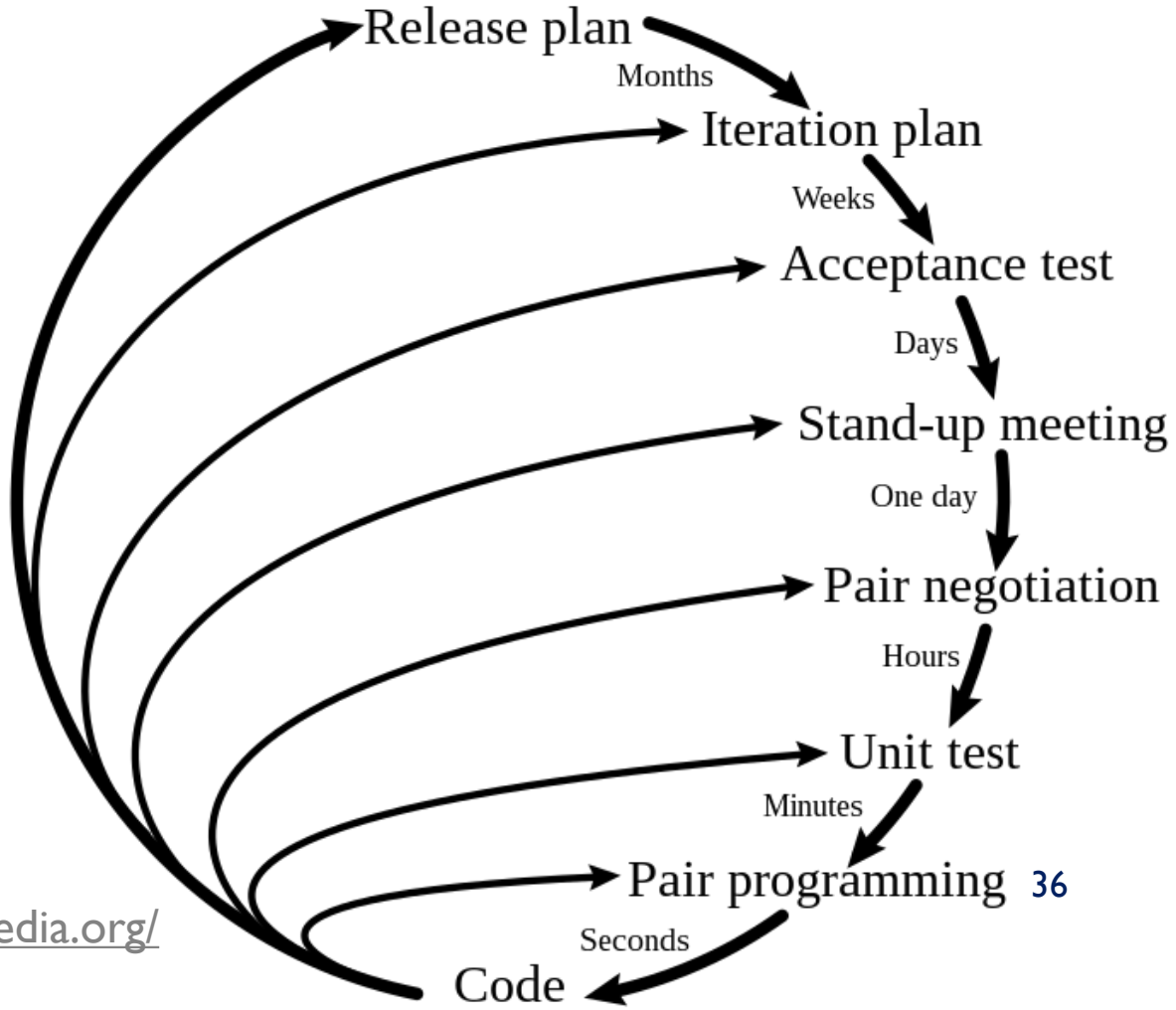
https://www.digite.com/

Kanban vs. Scrum

- Kanban is not time-boxed; so, no retrospective meeting is required because work is moving on all the time.
- Kanban has an easier workflow but Scrum is more structured and more forecasting is possible because the team velocity can be calculated to plan the next sprints.
- Kanban has less process so it's more flexible.
- Scrum can lead to scope creep if there is no end date set.
- Kanban is easily adaptable to large teams but Scrum works better in smaller teams.
- Change of the team structure in Scrum can mess up the velocity and planning.
- Daily standups exist in both frameworks.

Extreme Programming (XP)

- Extreme Programming is an Agile framework that is focused on fast release of high-quality software.
- XP is an iterative/incremental model.
- Elements of XP include:
  - Pair Programming and Mobbing
  - Spikes
  - Test First
  - Continuous Delivery
  - Monitoring System



https://www.wikipedia.org/

XP Practices: Pair Programming and Mobbing

- In XP every piece of code is written by the collaboration of two programmers working at the same station. While taking turns and sharing ideas, one programmer is the driver and types code and the other programmer is the navigator and gives ideas. This is called pair programming.
- Pairing in XP changes every day and people work with different people so everyone has almost equal expertise in the work being done.
- Mobbing or mob programming happens in bigger groups, may be 3 or 4 programmers, sometimes from different teams that write code at the same station. In mobbing one person is the driver and others are navigators. Every programmer has even time of being in the driver role.
- XP does not include any code review because code is written by more than one person and reviews will delay the work delivery.

XP Practices: Spikes

- To test something like a design idea before putting it into production work, some spikes might be implemented.
- Spikes are solo-done investigative work that include all the layers from front-end to back-end in which a small piece of each layer is implemented.
- Once the spike is implemented and the issue is understood the spike code is thrown away and people go back to pair programming.

XP Practices: Test First, Outside-In

- In XP before implementing any change, first a test is written to check if the intended behavior is not already implemented in the code base. Then the test is run and if the behavior has not been implemented then the change will be implemented.
- Test-first thinking also gives the team the confidence of refactoring: changing the code without breaking it.
- In XP also test is done outside-in; i.e. the acceptance test is written before unit test. The main purpose is to focus on what is really required to be implemented.

XP Practices: Continuous Delivery

- Another advantage of writing tests is that the code is always ready for deployment. The team never commits to any code that is not ready to deploy.
- Code commitments happen very frequently: multiple commits per day but the committed code is always ready to deploy.
- The work flow in XP includes a short pipeline with no staging environment and no manual testers.

XP Practices: Monitoring Systems

- Even though XP is test-driven, there are still some cases that are missed or are hard to test. That's why live automated monitoring systems are utilized that on a 24/7 basis they send automated messages to the team members informing them about if things are going as expected.

XP Practices: Refactoring

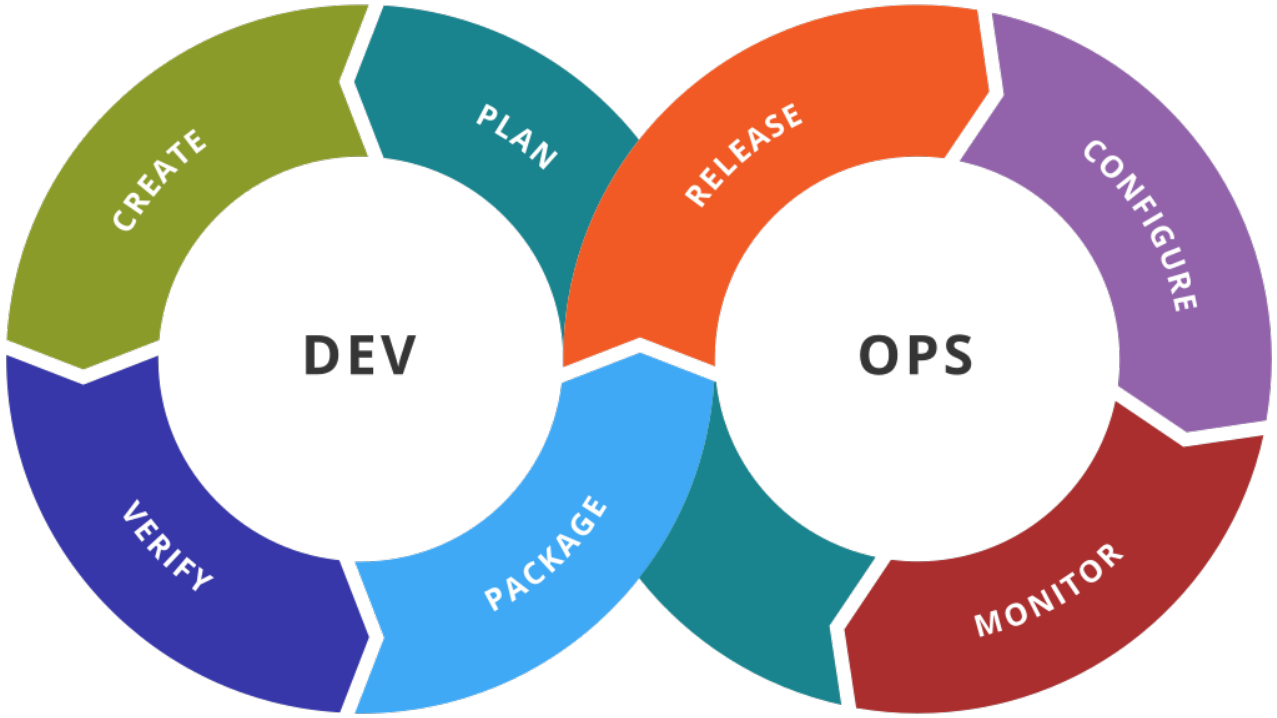
- One of the values of XP is that team members respect their own work by always striving for high quality and seeking for the best design for the solution at hand through refactoring.
- Refactoring means:
  - Changing the code structure without changing its functionality.
  - Changing the internal structure of the software without clients knowing about it.
  - Changingthe quality of code without changing its behavior.

Application Release Process

- How to deliver the application to the end users? (deployment)
  - Install tools
  - Deploy the application (build, test, etc.)
  - Configure the server
- After launching an application, it should be monitored for any probable bug or problem. In case of an error, it has to be fixed.
- After initial launch of the application still some improvements are applied: features are added, the server is improved, etc. Every update needs to be tracked and the software is versioned.
  - Software versioning: N.M.K where N changes if major changes happened, M shows minor changes, and K shows small changes like bug fixes or patches.

DevOps

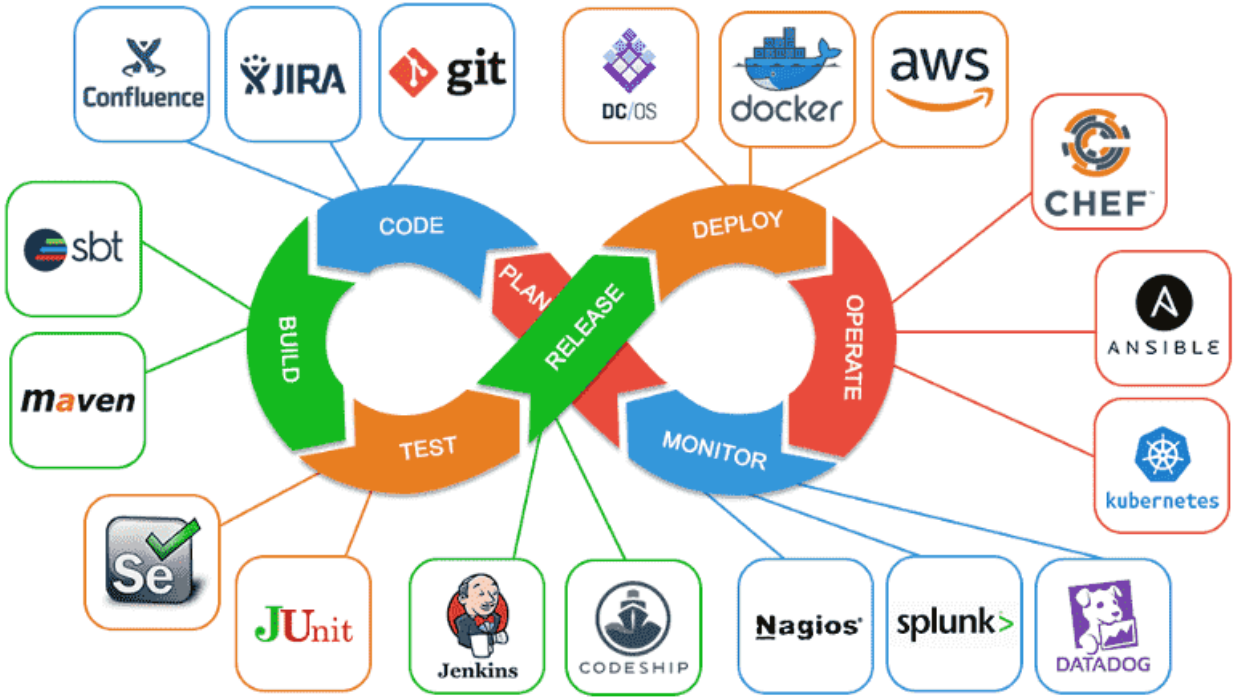
- DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the systems development life cycle and provide continuous delivery with high software quality.



https://www.wikipedia.org/

DevOps Toolchain

- A DevOps toolchain is a set or combination of tools that aid in the delivery, development, and management of software applications throughout the systems development life cycle.
- Generally, DevOps tools fit into one or more activities, which supports specific DevOps initiatives: Plan, Create, Verify, Package, Release, Configure, Monitor, ar Version Control.



https://www.techgeeknext.com/

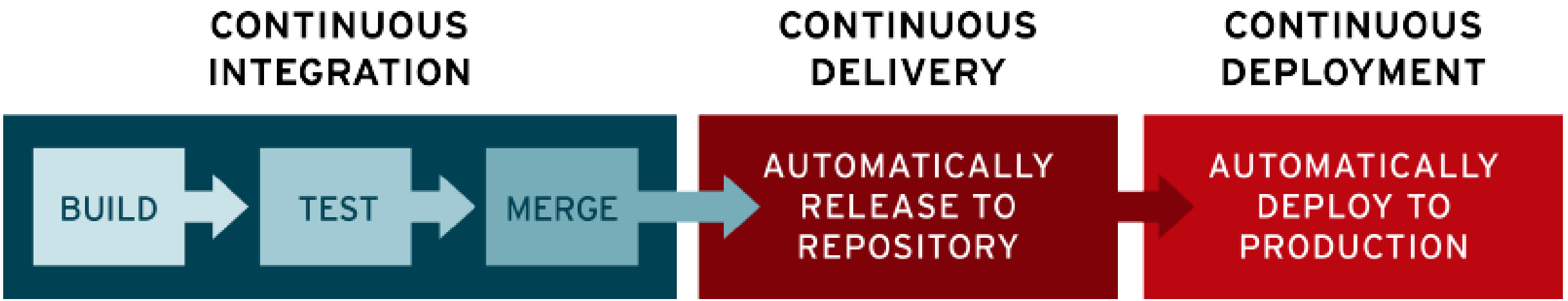
https://www.wikipedia.org/

CI/CD Pipeline

- A pipeline is a process that drives software development through a path of building, testing, and deploying code, also known as CI/CD.
- Tools that are included in the pipeline could include compiling code, unit tests, code analysis, security, and binaries creation.
- CI (Continuous Integration) is one of the core concepts of DevOps which means developers continuously commit their code into a shared repository. Each commit triggers an automated workflow on a CI Server that checks the integration for any conflict and notifies the developers if the integration failed or succeeded.
- CD (Continuous Delivery) means moving from one release every few months to multiple releases per day. CD automation tools ensure that high-quality software deployment happens more often and fast. Delivery is mainly automated but some manual operations like approvals may be involved.
- CD may refer to Continuous Deployment which means completely automated process.

https://www.redhat.com/

CI/CD Pipeline (Cont'd)



https://www.redhat.com/

Acknowledgements

- David Michel (*https://www.youtube.com/watch?v=20SdEYJEbrE*)
- Jeff Patton (*https://www.jpattonassociates.com/dont_know_what_i_want/*)
- Thomas Schissler (*https://www.scrum.org/thomas-schissler*)
- Mary Iqbal (https://www.rebelscrum.site/about)
- https://www.virtasant.com/
- https://www.indeed.com/
- https://www.javatpoint.com/
- https://www.geeksforgeeks.org/