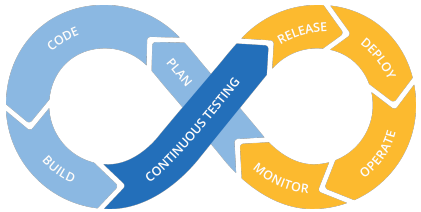


devops

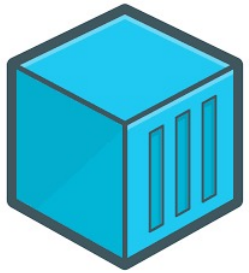
Git

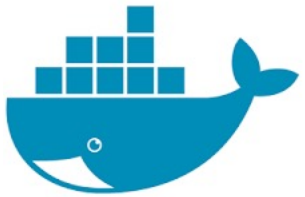
SE
Agile

Advanced Software Development Methodologies

Kubernetes

container





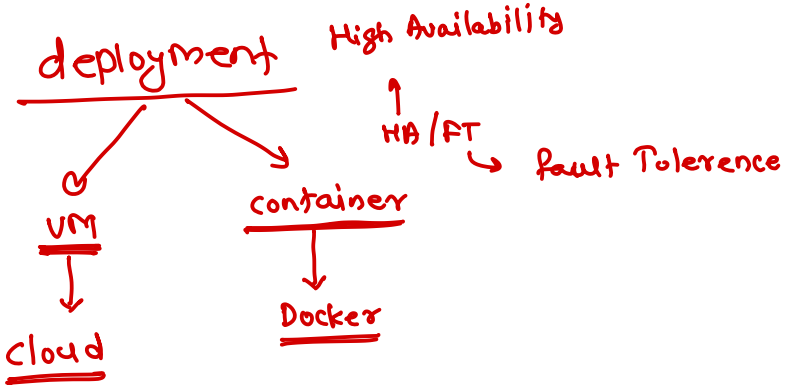
docker

Selenium

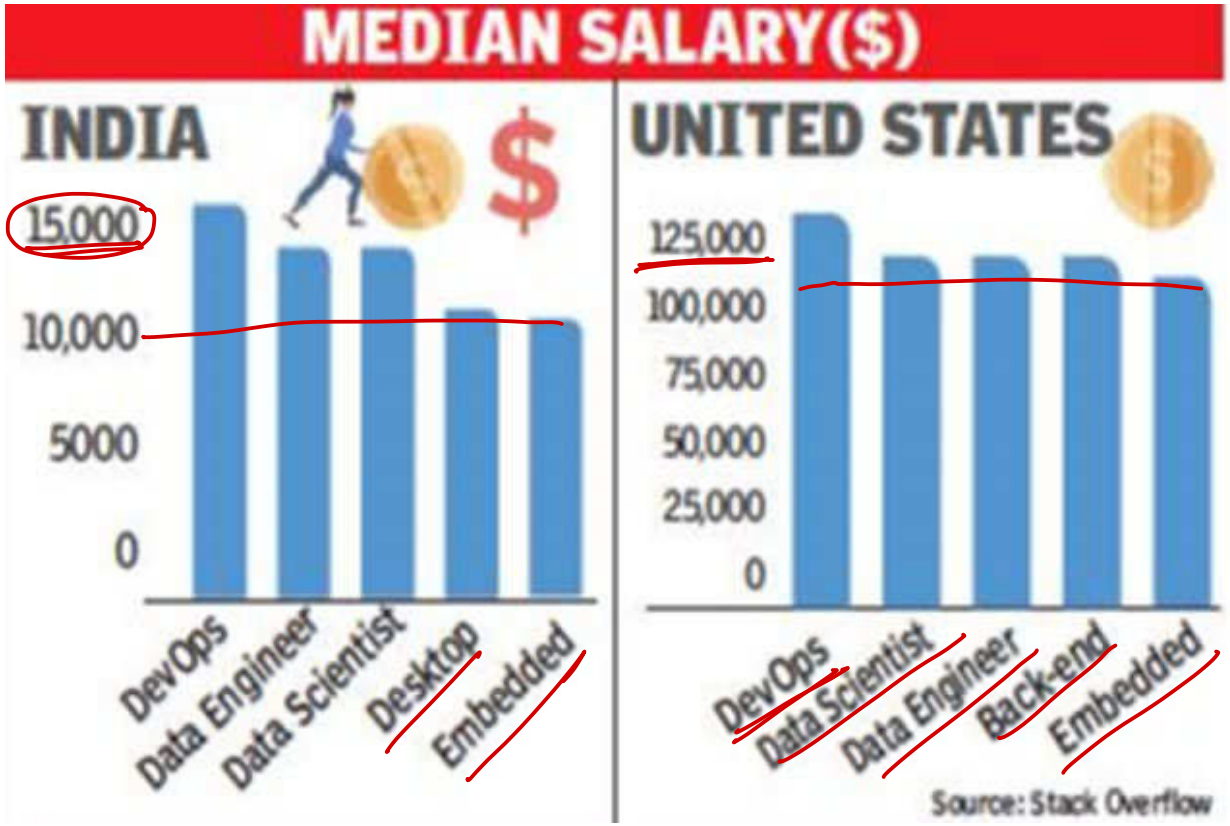
Jenkins

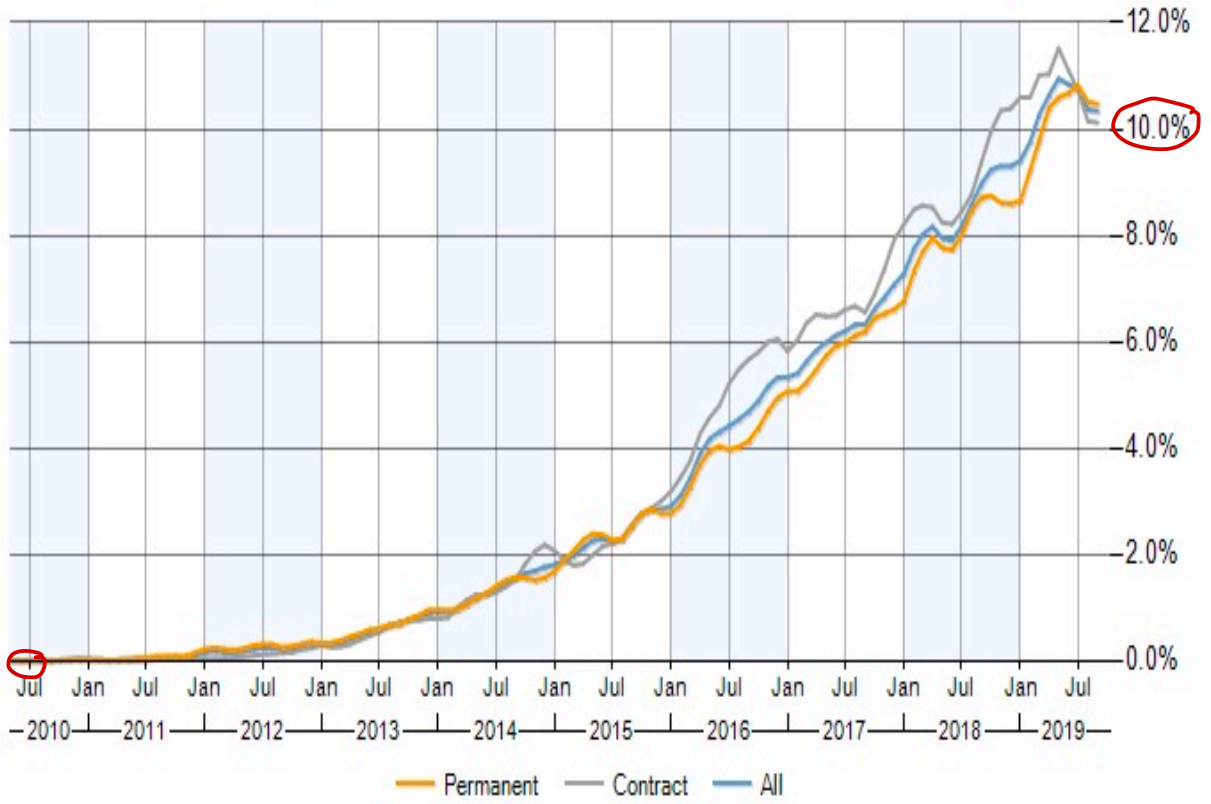
Contents

- Software Engineering
- SDLC VS Agile
- Application Testing
- Cloud Computing : AWS
- DevOps
- Source Control Management : Git → GitHub / GitLab
- Continuous Integration : Jenkins [CI/CD pipeline]
- Containerization : Docker
- Container Orchestration : Docker Swarm, Kubernetes
  - manage [create / restart / stop] containers



Why should you bother about DevOps?





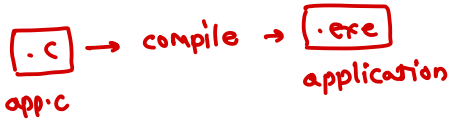
Pre-requisites

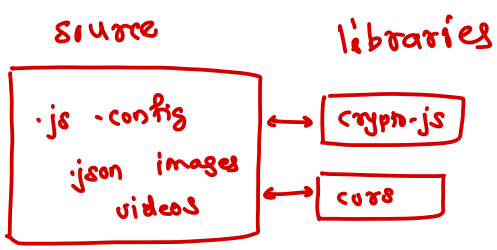
- Basic Linux knowledge
- Any Development Platforms → Node Js, Java
- Any language (preferred JS/Java)

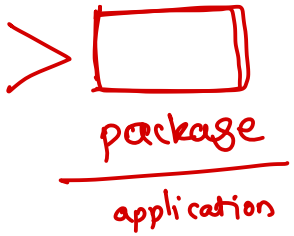
Software Engineering [SE]

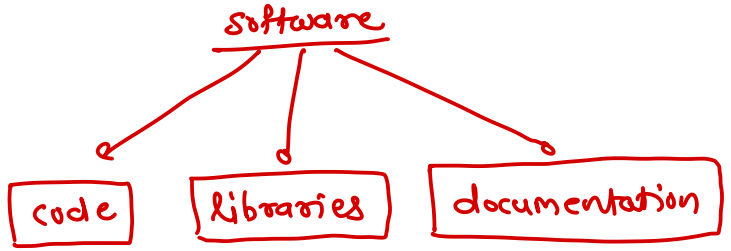
Introduction

- Software
  - Software is more than just a program code
  - A program is an executable code, which serves some computational purpose
  - Software is considered to be collection of executable programming code, associated libraries and documentations
  - Software, when made for a specific requirement is called software product
- Engineering
  - All about developing products using well defined principles, methods and procedures









What is Software Engineering?

- Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures
- The application of a systematic, disciplined , quantifiable approach to the development, operation and maintenance of software
- Establishment and use of sound engineering principles in order to obtain software that is reliable and work efficiently on real machines
- The process of developing a software product using software engineering principles and methods

Software Types

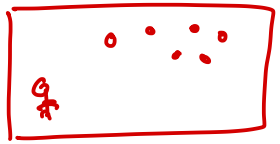
- System Software : operating System, Device Drivers
- Application Software : Calculator, Calendar, Email client
- Engineering/Scientific Software : Catia, Idea, AutoCAD,
- Embedded Software : IoT applications
- AI Software : Intrusion Detection System, Person detection in camera
- Legacy Software : old application
- Web/Mobile Software : Websites, mobile apps
  e-commerce

Why SE is important

- Helps to build complex systems in timely manner
- Ensures high quality of software & testing
- Imposes discipline to work
- Minimizes software cost
- Decreases time

Software Evolution

- **S-type (static-type)**
  - Works strictly according to the pre-defined specifications and solutions
  - Solution and method to achieve it can be understood immediately before coding starts
  - Least subjected to the changes
  - E.g. Calculator program for mathematical computation
- **P-type (practical-type)**
  - Software with a collection of different procedures
  - Is defined by exactly what procedures can do
  - The specification can be described and solutions are not obvious instantly
  - E.g. Gaming software
- **E-type (embedded-type)**
  - Works closely as the requirement of real-world environment
  - Has a high degree of evolution as there are various changes in laws, taxes etc. in the real world
  - E.g. online trading software

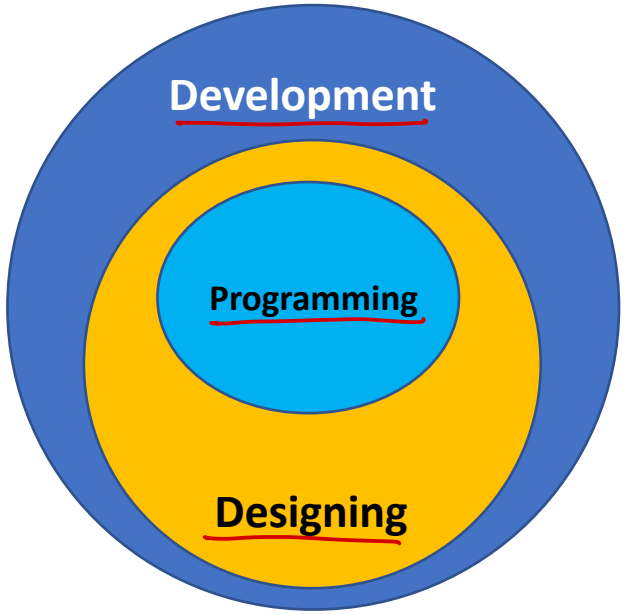


E-Type software evolution laws

- **Continuing change**
  - An E-type software system must continue to adapt to the real world changes, else it becomes progressively less useful
- **Increasing complexity**
  - As software evolves, its complexity tends to increase unless work is done to maintain or reduce it
- **Conservation of familiarity** → keep the documentation
  - The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc. must be retained at any cost, to implement the changes in the system
- **Continuing growth**
  - In order for an E-type system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business
- **Reducing quality**
  - An E-type software system declines in quality unless rigorously maintained and adapted to a changing operational environment
- **Feedback systems**
  - The E-type software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.
- **Self-regulation**
  - E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.
- **Organizational stability**
  - The average effective global activity rate in an evolving E-type system is invariant over the lifetime of the product.

Software Paradigms

- Refer to the methods and steps, which are taken while designing the software
- There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand
- These can be combined into various categories, though each of them is contained in one another
- Consists of
  - Requirement gathering
  - Software design
  - Programming



Characteristics of good software

- A software product can be judged by what it offers and how well it can be used
- Well-engineered and crafted software is expected to have the following characteristics
  1. Operational
  2. Transactional
  3. Maintenance

Operational

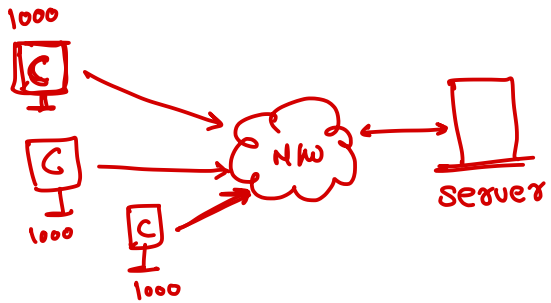
- This tells us how well software works in operations
- Can be measured on:
  - Budget
  - Usability
  - Efficiently
  - Correctness
  - Functionality
  - Dependability
  - Security : confidentiality [encryption]
  - Safety : parental control

Transitional

- This aspect is important when the software is moved from one platform to another
- Can be measured on
  - Portability
  - Interoperability
  - Reusability
  - Adaptability

Maintenance

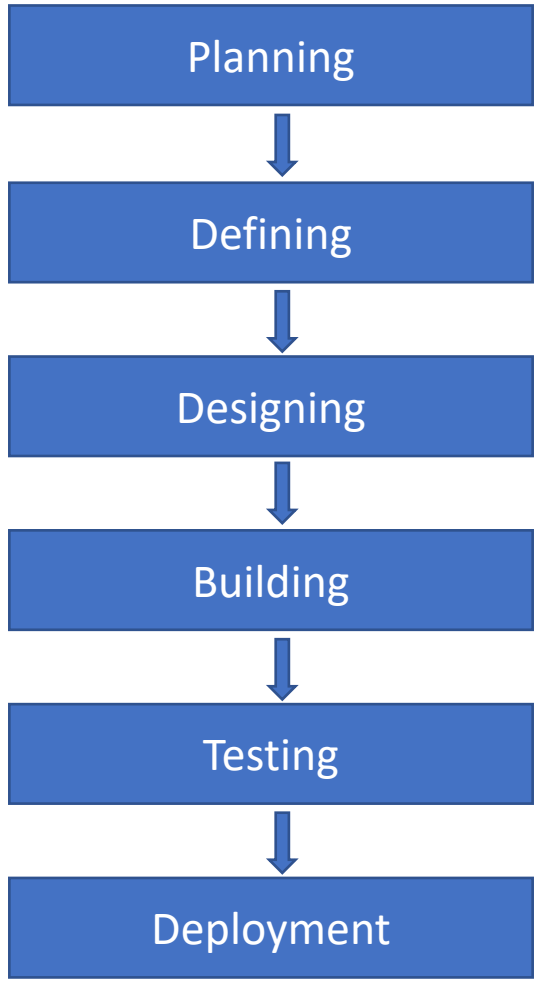
- Briefs about how well a software has the capabilities to maintain itself in the ever-changing environment
- Can be measured on
  - Modularity
  - Maintainability
  - Flexibility
  - Scalability
    - handling as much load as possible
    - solutions
      - vertical
      - horizatal




SDLC

Overview

- Also called as Software Development Process
- Is a well-defined, structured sequence of stages in software engineering to develop the intended software product
- Is a framework defining tasks performed at each step in the software development process
- Aims to produce a high-quality software that
  - meets or exceeds customer expectations
  - reaches completion within times and cost estimates
- Consists of detailed plan (stages) of describing how to develop, test, deploy and maintain the software product



Planning and Requirement Analysis

- The most important and fundamental stage in SDLC
- It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry
- This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas
- Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage
- The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks

Defining Requirements

- Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts
- This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle

Designing the Product Architecture

- SRS is the reference for product architects to come out with the best architecture for the product to be developed
- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification
- This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product
- A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any)
- The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS

Building or Developing the Product

- In this stage of SDLC the actual development starts and the product is built
- The programming code is generated as per DDS during this stage
- If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle
- Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code
- Different high level programming languages such as C, C++, Java, PHP etc. are used for coding
- The programming language is chosen with respect to the type of software being developed

Testing the Product

- This stage is usually a subset of all the stages as in the modern SDLC models
- The testing activities are mostly involved in all the stages of SDLC
- However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS

Deployment in the Market and Maintenance

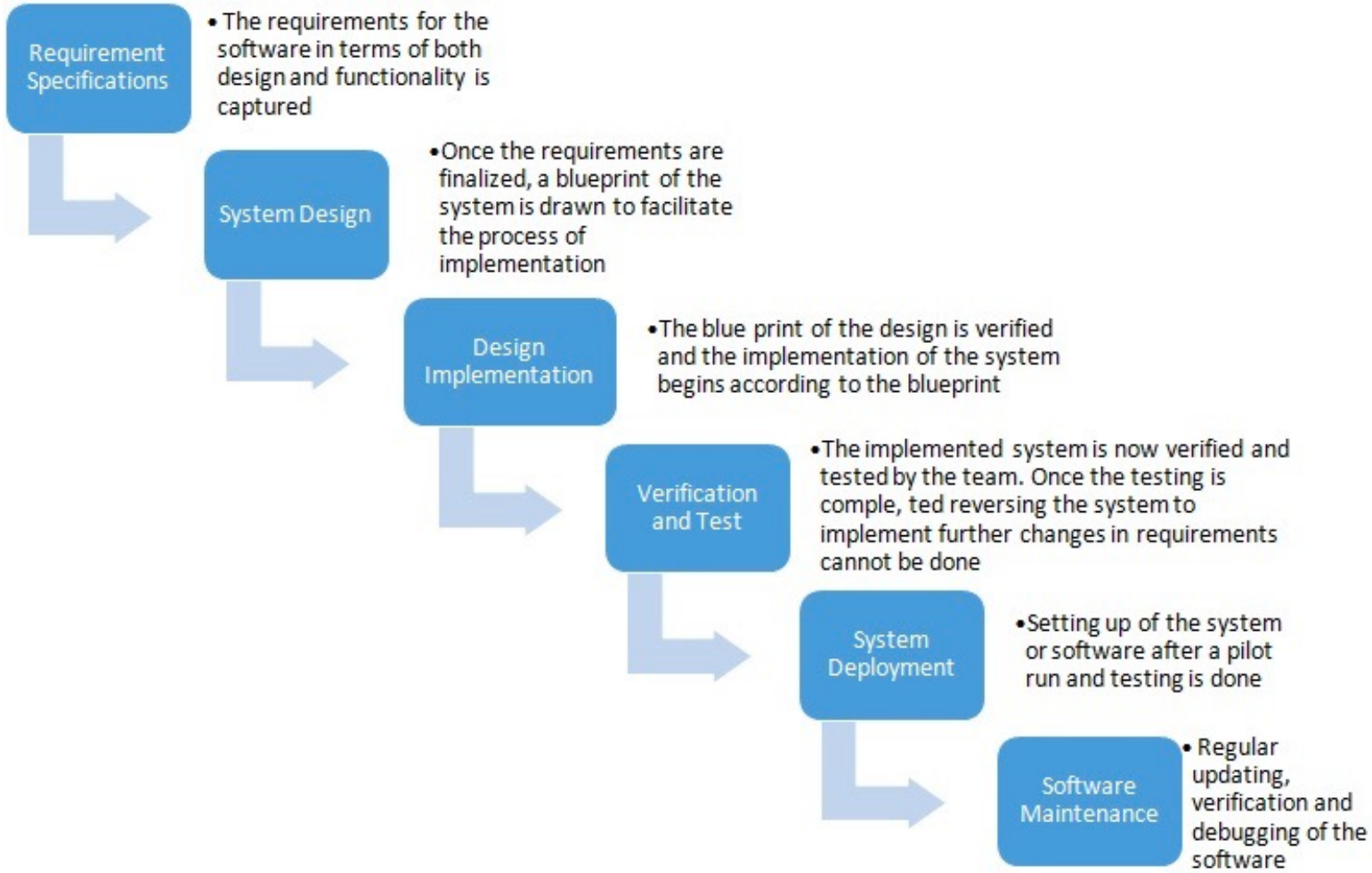
- Once the product is tested and ready to be deployed it is released formally in the appropriate market
- Sometimes product deployment happens in stages as per the business strategy of that organization
- The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing)
- Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment
- After the product is released in the market, its maintenance is done for the existing customer base

SDLC Models

- There are various software development life cycle models defined and designed which are followed during the software development process
- Also referred as Software Development Process Models
- Models
  - Waterfall Model
  - Iterative Model
  - Spiral Model
  - V-Model
  - Big Bang Model
  - Agile Model

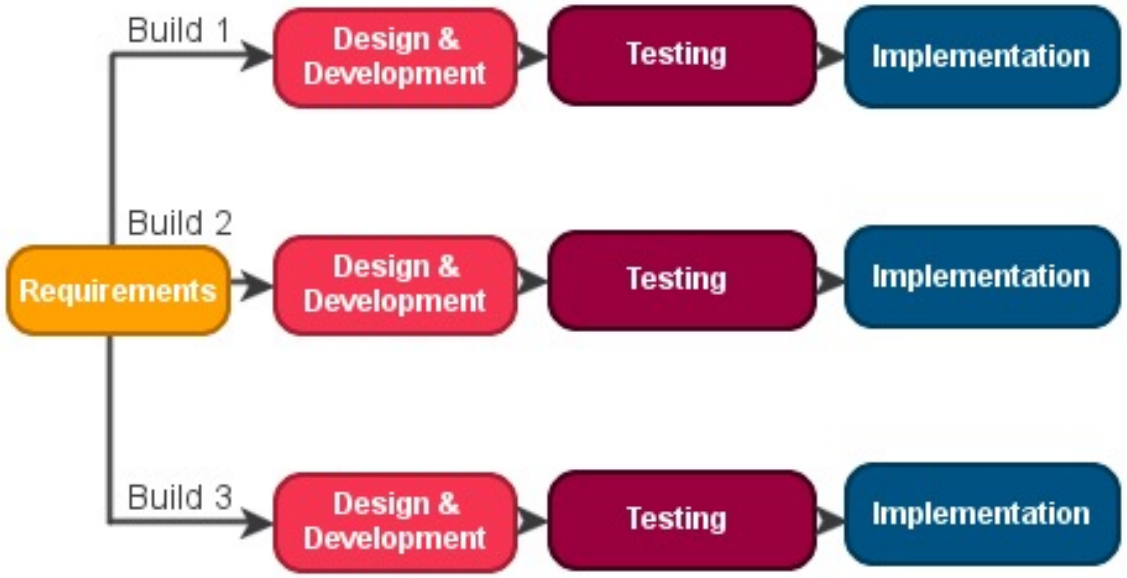
Waterfall Model

- **Requirement Specification**
- **Design**
- **Implementation**
- **Verification and Testing**
- **Deployment**
- **Maintenance**



Iterative Model

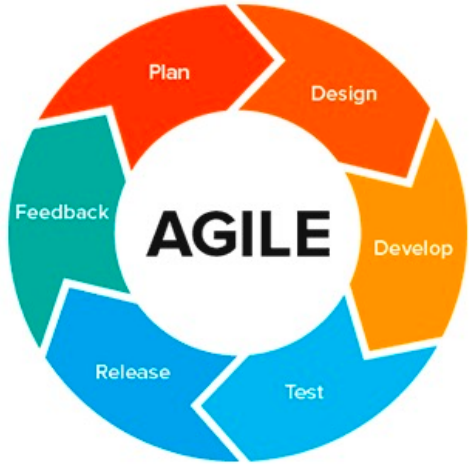
- implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented



Agile Methodologies

Agile Methodologies

- Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements
- The tasks are divided to time boxes (small time frames) to deliver specific features for a release
- Iterative approach is taken and working software build is delivered after each iteration
- Each build is incremental in terms of features; the final build holds all the features required by the customer



Agile Manifesto

- **Individuals and interactions**
  - self-organization and motivation are important
- **Working software**
  - Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentations
- **Customer collaboration**
  - continuous customer interaction is very important to get proper product requirements
- **Responding to change**
  - focused on quick responses to change and continuous development

Agile Methodologies

- The most popular Agile methods include
  - Rational Unified Process
  - Scrum
  - Crystal Clear
  - Extreme Programming
  - Adaptive Software Development
  - Feature Driven Development
  - Dynamic Systems Development Method (DSDM)

What is Scrum ?

- Scrum isn't a process, it's a framework that facilitates processes amongst other things
- Is an agile way to manage a project
- Management framework with far reaching abilities to control and manage the iterations and increments in all project types
- One of the implementations of agile methodology
- Incremental builds are delivered to the customer in every two to three weeks time
- Ideally used in the project where the requirement is rapidly changing
- The framework is made up of a Scrum team, individual roles, events, artefacts, and rules

Scrum Principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- Welcome changing requirements, even late in development
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
- Business people and developers must work together daily throughout the project
- Build projects around motivated individuals
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

Scrum Principles

- Working software is the primary measure of progress
- Agile processes promote sustainable development
- Continuous attention to technical excellence and good design enhances agility
- Simplicity, the art of maximizing the amount of work not done, is essential
- The best architectures, requirements, and designs emerge from self-organizing teams
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Scrum Pillars

Transparency

All parts of the process should be transparent, open, and honest for everyone to see

Inspection

Everything that is worked on during a sprint can be inspected by the team to make sure that it is achieving what it needs to.

Adaptation

If a member of the Scrum team or a stakeholder notices that things aren't going according to plan, the team will need to change up what they're doing to fix this as quickly as possible

Scrum Values

Courage	Courage to do the right thing and work on tough problems
Focus	Focus on the sprint and its goal
Commitment	Commitment to the team and sprint goal
Respect	Respect, for each other by helping people to learn the things that you're good at, and not judging the things that others aren't good at
Openness	Be open and honest and let people know what you're struggling with challenges and problems that are stopping you from achieving success

How scrum pillars help us?

Self-organization

- This results in healthier shared ownership among the team members
- It is also an innovative and creative environment which is conducive to growth

Collaboration

- Essential principle which focuses collaborative work

Time-boxing

- Defines how time is a limiting constraint in Scrum method
- Daily Sprint planning and Review Meetings

Iterative Development

- Emphasizes how to manage changes better and build products which satisfy customer needs
- Defines the organization's responsibilities regarding iterative development

Scrum Roles

Product Owner

- Job to understand and engage with the stakeholders to understand what needs to be done and create that backlog
- Also need to prioritize that backlog

Scrum Master

- Helps the entire team achieve the scrum goals and work within scrum
- Support the product owner with their responsibilities in terms of managing the backlog as well as, supporting the development team

Dev Team

- The people who are creating the product or service and delivering done increments at the end of each sprint
- Includes developers, tester, writers, graphics artists and others

Scrum Artefacts

Product Backlog

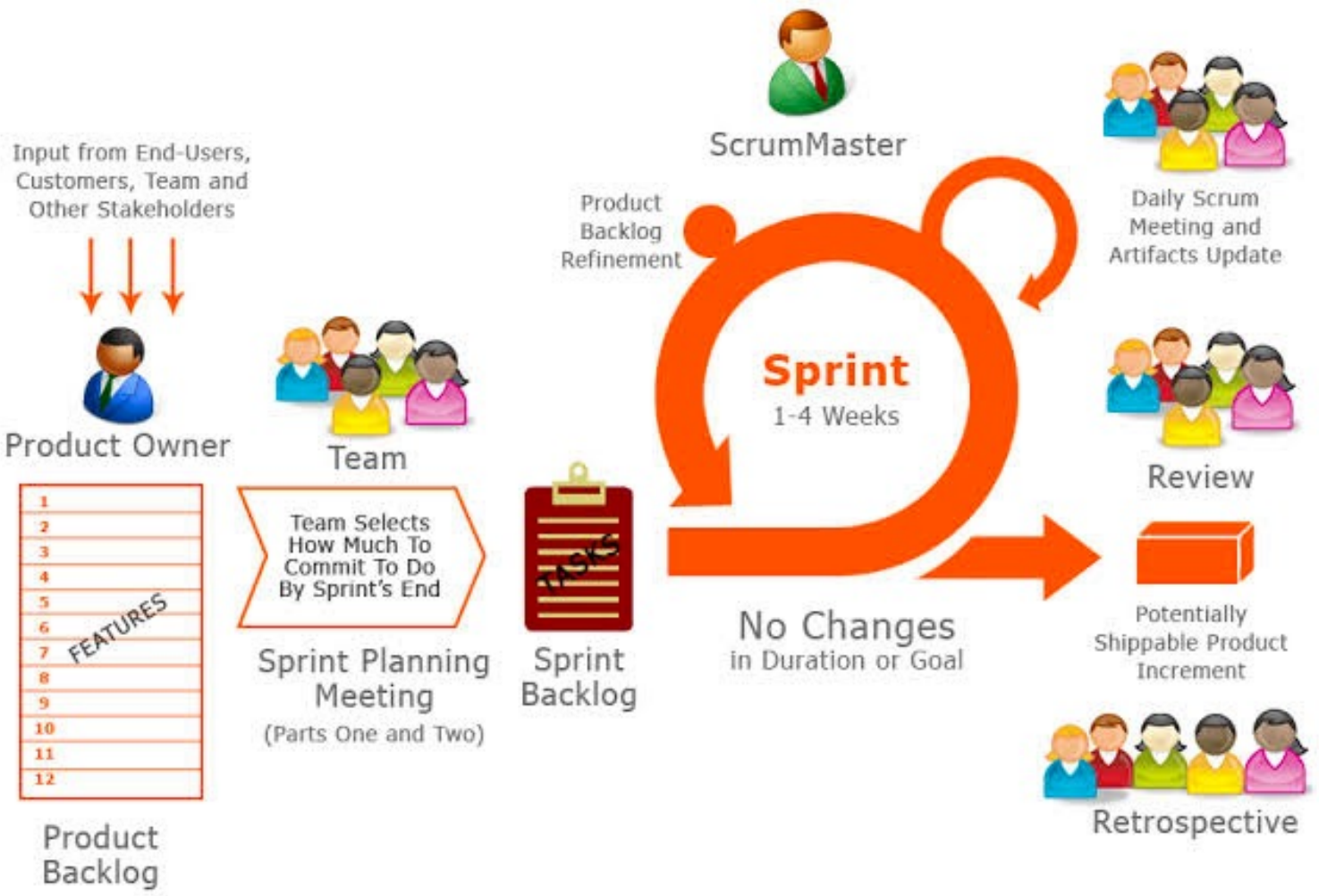
Sprint Backlog

Increment

Scrum Events

Sprint Planning	• Happens at the start of every sprint • it should probably be about between four and eight hours
Daily Scrum	• This is a very short time-boxed event • Usually only lasting no more than 15 minutes
Sprint Review	• Collaborative events to demo what has been achieved and to help keep everyone who's involved working together
Sprint Retrospective	• About continuous process and improvement and we need to take what we've learned into the next sprint planning session
Sprint	• It is really the beating heart of scrum and all the scrum events take place in it

Scrum Process



Agile vs traditional models

No	Agile Methodologies	Traditional Methodologies
1	Incremental value and risk management	Phased approach with an attempt to know everything at the start
2	Embracing change	Change prevention
3	Deliver early, fail early	Deliver at the end, fail at the end
4	Transparency	Detailed planning, stagnant control
5	Inspect and adapt	Meta solutions, tightly controlled procedures and final answers
6	Self managed	Command and control
7	Continual learning	Learning is secondary to the pressure of delivery

Agile - Advantages

- Very realistic approach to software development
- Promotes teamwork and cross training
- Functionality can be developed rapidly and demonstrated
- Resource requirements are minimum
- Suitable for fixed or changing requirements
- Delivers early partial working solutions
- Good model for environments that change steadily
- Minimal rules, documentation easily employed
- Enables concurrent development and delivery within an overall planned context
- Little or no planning required
- Easy to manage
- Gives flexibility to developers

Agile - Disadvantages

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

Scrum tools

- Jira – https://www.atlassian.com/software/jira/
- Clarizen – https://www.clarizen.com/
- GitScrum – https://site.gitscrum.com/
- Vivify Scrum – https://www.vivifyscrum.com/
- Yodiz – https://www.yodiz.com/
- ScrumDo – https://www.scrumdo.com/
- Quickscrum – https://www.quickscrum.com/
- Manuscript – https://www.manuscript.com/
- Scrumwise – https://www.scrumwise.com/
- Axosoft – https://www.axosoft.com/

Agile Methodologies – Scrum Terminologies

- **Scrum:** a framework to support teams in complex product development
- **Scrum Board:** a physical board to visualize information for and by the Scrum Team, used to manage Sprint Backlog
- **Scrum Master:** the role within a Scrum Team accountable for guiding, coaching, teaching and assisting a Scrum Team and its environments in a proper understanding and use of Scrum
- **Scrum Team:** a self-organizing team consisting of a Product Owner, Development Team and Scrum Master
- **Self-organization:** the management principle that teams autonomously organize their work

Agile Methodologies – Scrum Terminologies

- **Sprint:** time-boxed event of 30 days, or less, that serves as a container for the other Scrum events and activities.
- **Sprint Backlog:** an overview of the development work to realize a Sprint's goal, typically a forecast of functionality and the work needed to deliver that functionality.
- **Sprint Goal:** a short expression of the purpose of a Sprint, often a business problem that is addressed
- **Sprint Retrospective:** time-boxed event of 3 hours, or less, to end a Sprint to inspect the past Sprint and plan for improvements

Agile Methodologies – Scrum Terminologies

- **Sprint Review:** time-boxed event of 4 hours, or less, to conclude the development work of a Sprint
- **Stakeholder:** a person external to the Scrum Team with a specific interest in and knowledge of a product that is required for incremental discovery
- **Development Team:** the role within a Scrum Team accountable for managing, organizing and doing all development work
- **Daily Scrum:** daily time-boxed event of 15 minutes for the Development Team to re-plan the next day of development work during a Sprint