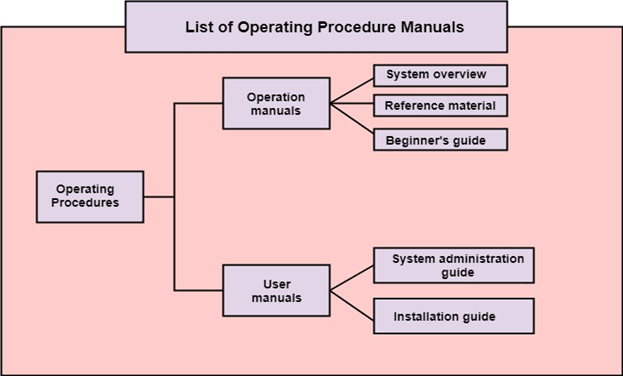
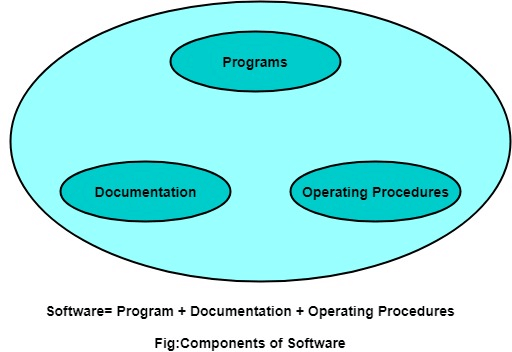
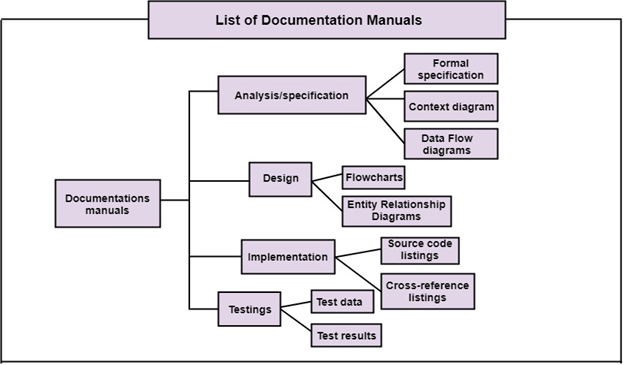
**SOFTWARE ENGINEERING**

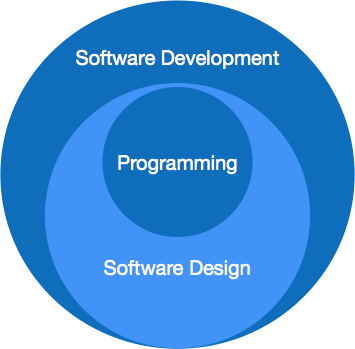
**Introduction to software engineering**

* The term software specifies to the set of computer programs, procedures and associated documents (Flowcharts, manuals, etc.) that describe the program and how they are to be used.
* Software Engineering is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.
* Software Engineering is a systematic, disciplined, quantifiable study and approach to the design, development, operation, and maintenance of a software system.

**Software Process**

A software process is the set of activities and associated outcome to produce a software product. There are four key process activities

1. **Software specifications:** The functionality of the software and constraints on its operation must be defined.
2. **Software development:** The software to meet the requirement must be produced.
3. **Software validation:** The software must be validated to ensure that it does what the customer wants.
4. **Software evolution:** The software must evolve to meet changing client needs.

Software paradigms (process or model) refer to the methods and steps, which are taken while designing the software. These can be combined into various categories, though each of them is contained in one another.

**Programming paradigm** is a subset of Software design paradigm which is further a subset of Software development paradigm.

**Software Development Paradigm:** This Paradigm is known as software engineering paradigms where all the engineering concepts pertaining to the development of software are applied. It includes various researches and requirement gathering which helps the software product to build. It consists of –

1. Requirement gathering
2. Software design
3. Programming

**Software Design Paradigm:** This paradigm is a part of Software Development and includes –

1. Design
2. Maintenance
3. Programming

**Software Process Model:**

To produce a software product the set of activities is used. This set is called a software process. Models are a simplification, so a software process model is an abstraction of the actual process, which is being described. Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering.

Basic software process models on which different type of software process models can be implemented:

**A workflow Model** –It is the sequential series of tasks and decisions that make up a business process.

**The Waterfall Model –** It is a sequential design process in which progress is seen as flowing steadily downwards. Phases in waterfall model:

1. Requirements Specification
2. Software Design
3. Implementation
4. Testing

**Dataflow Model –** It is diagrammatic representation of the flow and exchange of information within a system.

**Evolutionary Development Model –** Following activities are considered in this method:

1. Specification
2. Development
3. Validation

**Role / Action Model –** Roles of the people involved in the software process and the activities.

**Software Product**

**Characteristics of Well-Engineered Software Product:**

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds:

* Operational
* Transitional
* Maintenance

**Operational:** This tells us how well software works in operations. It can be measured on:

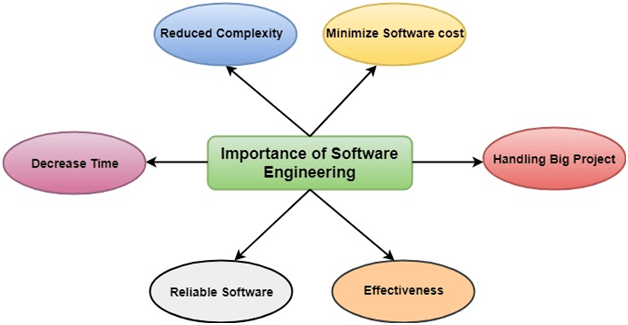
* Budget
* Usability
* Efficiency
* Correctness
* Functionality
* Dependability
* Security
* Safety

**Transitional:** is important when the software is moved from one platform to another:

* Portability
* Interoperability
* Reusability
* Adaptability

**Maintenance:** This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment:

* Modularity
* Maintainability
* Flexibility
* Scalability

**Importance of Software engineering**

Software Engineering is required due to the following reasons:

* To manage Large software
* For more Scalability
* Cost Management
* To manage the dynamic nature of software
* For better quality Management

**Software Development Life Cycles:**

* Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality soft-wares.
* Also called as Software Development Process.
* SDLC is a framework defining tasks performed at each step in the software development process.
* ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

A typical Software Development Life Cycle consists of the following stages −

**Stage 1: Planning and Requirement Analysis**

* Performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.
* Planning for the quality assurance requirements and identification of the risks associated with the project
* This information is then used to plan the basic project approach and to conduct ***product feasibility study*** in the economical, operational and technical areas. 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.

**Stage 2: Defining Requirements**

* 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.

**Stage 3: Designing the Product Architecture**

* SRS is the reference for product architects to come out with the best architecture for the product to be developed.
* 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 like 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.
* The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

**Stage 4: Building or Developing the Product**

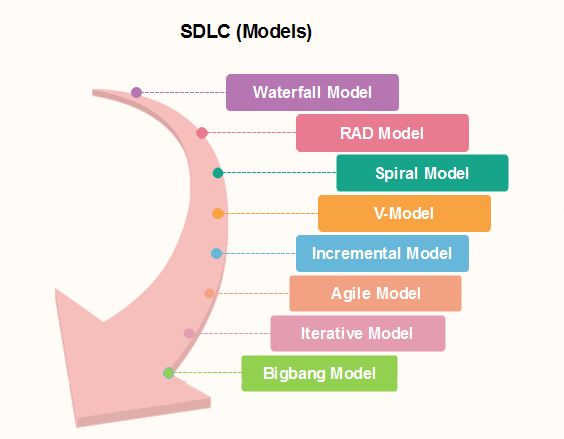
* In this stage of SDLC the actual development starts and the product is built.
* The programming code is generated as per DDS during this stage
* Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. The programming language is chosen with respect to the type of software being developed.

**Stage 5: Testing the Product**

* This stage is usually a subset of all the stages as in the modern SDLC models
* 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.

**Stage 6: 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 followed in the industry**

**Adaptive Software Development (ASD):** is a method to build complex software and system. ASD focuses on human collaboration and self-organization. ASD “life cycle” incorporates three phases namely: 1. Speculation 2. Collaboration 3. Learning

People working together must trust

**4GT Model:** The term fourth generation techniques (4GT) encompass a broad array of software tools that have one thing in common: each enables the software engineer to specify some characteristic of software at a high level. The tool then automatically generates source code based on the developer's specification.

**COCOMO model:** (Constructive Cost Model) is a regression model based on LOC, i.e number of Lines of Code. It is a procedural cost estimate model that is frequently used as a method of accurately estimating size, effort, cost, duration, and quality for a project completion.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Classical Waterfall (Linear) | Spiral | V Shaped(Verification/Validation) |
|  | The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.  Linear ordering of activities has some significant consequences. First, to identify the end of a phase and the beginning of the next, some certification techniques have to be employed at the end of each step. Some verification and validation usually do this mean that will ensure that the output of the stage is consistent with its input (which is the output of the previous step), and that the output of the stage is consistent with the overall requirements of the system. | The spiral model is a risk-driven process model. The spiral technique is a combination of rapid prototyping and concurrency in design and development activities. Size of ring speaks about the cost,  Each cycle in the spiral begins with the identification of objectives for that cycle, the different alternatives that are possible for achieving the goals, and the constraints that exist. This is the first quadrant of the cycle (upper-left quadrant).  The next step in the cycle is to evaluate these different alternatives based on the objectives and constraints. The focus of evaluation in this step is based on the risk perception for the project.  The next step is to develop strategies that solve uncertainties and risks. This step may involve activities such as benchmarking, simulation, and prototyping. | In this type of SDLC model testing and the development, the step is planned in parallel. So, there are verification phases on the side and the validation phase on the other side.  V model and waterfall model are the same except that the test planning and testing start at an early stage in V-Model. |
| **Characteristics** | Static technology that is understood  Stable product definition  Strictly-set long timeline  Fixed budget that cannot be changed | Flexibility and risk handling  From the Iterative methodology, it took the cycled development, and from the Waterfall one, it borrowed a systematic approach and the ability to control the process | every stage of development includes testing as well.  the development phase corresponds to the testing phase and they go parallelly |
| **Suitable scenario** | Basic model for all life cycle models  Short-term projects | medium or high-risk project, long-term project, and changes might occur. The requirements are vague to the customer, product to be released in several stages for feedback | There are clearly defined, documented, and stable requirements  All of the requirements are extra clear  The project is short-term and not complex. best for healthcare industry |
| **Advantages** | Well documented  Waterfall model is the simple model which can be easily understood and is the one in which all the phases are done step by step.  Deliverables of each phase are well defined, and this leads to no complexity and makes the project easily manageable. | can add elements to the product in case they become available  Risk Analysis is done extensively using the prototype models.  Any enhancement or change in the functionality can be done in the next iteration. | It is a simple and easily understandable model.  V –model approach is good for smaller projects wherein the requirement is defined and it freezes in the early stage.  It is a systematic and disciplined model which results in a high-quality product. |
| **Disadvantages** | No mechanism for error correction (found during a different phase of development)  Hard to go back and change its conceptual basics or documentation  time-consuming & cannot be used in the short duration projects  cannot be used for the projects which have uncertain requirement or | The spiral model is best suited for large projects only.  The cost can be high as it might take a large number of iterations which can lead to high time to reach the final product. | V-shaped model is not good for ongoing projects.  Requirement change at the later stage would cost too high. |

|  |  |  |
| --- | --- | --- |
|  | Agile | Prototype Model |
|  | Agile methodology is a practice which promotes continues interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.  Any agile software phase is characterized in a manner that addresses several key assumptions about the bulk of software projects:  It is difficult to think in advance which software requirements will persist and which will change. It is equally difficult to predict how user priorities will change as the project proceeds.  For many types of software, design and development are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to think about how much design is necessary before construction is used to test the configuration.  Analysis, design, development, and testing are not as predictable (from a planning point of view) as we might like. designed to incorporate change requests quickly  to deliver an increment to the customer after each Time-box. The end date of an iteration is fixed, it can’t be extended | The prototyping model starts with the requirements gathering. The developer and the user meet and define the purpose of the software, identify the needs, etc.  A 'quick design' is then created. This design focuses on those aspects of the software that will be visible to the user. It then leads to the development of a prototype. The customer then checks the prototype, and any modifications or changes that are needed are made to the prototype.  Looping takes place in this step, and better versions of the prototype are created. These are continuously shown to the user so that any new changes can be updated in the prototype. This process continue until the customer is satisfied with the system. Once a user is satisfied, the prototype is converted to the actual system with all considerations for quality and security.  Prototype models have limited functional capabilities and inefficient performance when compared to the actual software. This is a valuable mechanism for understanding the customers’ needs. |
| **Suitable scenario** | agility is achieved by removing unnecessary activities that waste time and effort  It allows more flexibility to adapt to the changes.  The new feature can be added easily.  Customer satisfaction as the feedback and suggestions are taken at every stage.  small services projects | Prototype model reduces the cost and time of development as the defects are found much earlier.  Missing feature or functionality or a change in requirement can be identified in the evaluation phase and can be implemented in the refined prototype.  Involvement of a customer from the initial stage reduces any confusion in the requirement or understanding of any functionality. |
| **Advantages** | Lack of documentation.  Agile needs experienced and highly skilled resources.  If a customer is not clear about how exactly they want the product to be, then the project would fail. | Since the customer is involved in every phase, the customer can change the requirement of the end product which increases the complexity of the scope and may increase the delivery time of the product. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Incremental Model | Iterative Model | Big bang model | Rapid Application Development |
| **Summery** | The incremental model is not a separate model. It is necessarily a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, each cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete. | It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.  Phases of Iterative & Incremental Development:  Inception phase  Elaboration Phase  Construction Phase  Transition Phase | Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.  This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements are either unknown or final release date is not given. | RAD is an adoption of the waterfall model; it targets developing software in a short period. The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.  Business Modeling  Data Modeling  Process Modeling  Application Generation  Testing and Turnover |
| **Advantages** |  | Any change in the requirement can be easily done and would not cost as there is a scope of incorporating the new requirement in the next iteration.  Risk is analyzed & identified in the iterations.  Defects are detected at an early stage.  As the product is divided into smaller chunks it is easy to manage the product. | It’s a very simple Model.  Less Planning and scheduling is required.  The developer has the flexibility to build the software of their own. |  |
| **Disadvantages** |  | Complete requirement and understanding of a product are required to break down and build incrementally. | Big Bang models cannot be used for large, ongoing & complex projects.  High risk and uncertainty. |  |

**Four Core values of Agile Framework.**

1. Individuals and interactions over processes and tools. ...
2. Working software over comprehensive documentation. ...
3. Customer collaboration over contract negotiation. ...
4. Responding to change over following a plan.

**Key Agile components:** User story, sprint, daily scrum meeting, Kanban board (work-flow), product backlog.

**Agile Methodology Phases:** Build the Product Backlog, Sprint planning and sprint backlog, Sprint in progress, Beta testing and Product Demo, Sprint Review and Retrospective

**8 Important Types Of Agile Methodology:**

1. **Kanban:** This method uses visual methods for developing and managing projects. Kanban Board, which is divided into columns to depict the process flow of the software development.
2. **Scrum** phases of effective project management · 1. Initiation · 2. Planning and estimation · 3. Implementation · 4. Reviewing · 5. Releasing. (actually 3 A Product Owner orders the work for a complex problem into a Product Backlog.The Scrum Team turns a selection of the work into an Increment of value during a Sprint.The Scrum Team and its stakeholders inspect the results and adjust for the next Sprint.)
3. **Extreme Programming (XP):** Extreme Programming (XP) is a methodology that emphasizes teamwork, communication, and feedback. It focuses on constant development and customer satisfaction. Similar to scrum, this method also uses sprints or short development cycles. In Extreme Programming, the project is tested from the initial stages by collecting feedback that progresses the output of the system. This also presents a spot check to implement easily any customer requirements.
4. **Crystal:** Crystal is a group of smaller agile development methodologies comprising Crystal Yellow, Crystal Clear, Crystal Red, Crystal Orange, and more. Each has its peculiar and exclusive framework that is characterized by factors such as system criticality, team size, and project priorities. Depending on the nature of the project or system criticality such as Comfort (C), Essential Money (E), Discretionary Money (D), and Life (L), the kind of crystal agile methodology is chosen. The Crystal family advocates that each system or project is inimitable and necessitates the solicitation of diverse practices, processes, and policies to achieve the best results, earning the name of the most lightweight methods of agile methodology.
5. **Dynamic Systems Development Method (DSDM):** charter for the swift delivery of software. The DSDM believes that modifications to the project are always expected, and quality with timely delivery must never be negotiated.
6. Feature-Driven Development (FDD): Its primary goal is to consistently produce working software in a timely fashion. Lifecycle stages include developing an overarching model of the project; creating feature lists; planning by feature; designing by feature; and finally building by feature.
7. **Lean Software Development:** This agile methodology is based on seven principles:
   1. Deleting what doesn’t matter
   2. Quality development- The discipline and control of the number of residuals created are essential to quality development
   3. Knowledge creation- The team is driven to document the entire infrastructure to preserve this value in the future
   4. Defer commitments- focus less on planning and anticipating ideas without first having a prior and complete understanding of the business requirements
   5. Delivery promptly- Providing value to the customer as quickly as possible
   6. Respecting the team- two essential points are communication and conflict management
   7. Optimize the whole- To create a flow of true value, the development sequence must be perfected enough to remove errors from the code
8. **Scaled Agile Framework (SAFe):** Adopting SAFe allows you to take advantage of a framework that is relatively light while still maintaining the centralized decision-making required at the enterprise level for software development efficiency.

**Requirements Engineering**

Requirements engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process.

FAST stands for Facilitated Application Specification Technique

**Types of Requirements**

**Functional requirements:** These are the requirements that the end user specifically demands as basic facilities that the system should offer. These are represented or stated in the form of input to be given to the system, the operation performed, and the output expected.

**Non-functional requirements / Non-Behavioral Requirements:** These are basically the quality constraints that the system must satisfy according to the project contract.

1. Portability
2. Security
3. Maintainability
4. Reliability
5. Scalability
6. Performance
7. Reusability
8. Flexibility

NFR’s are classified into following types:

1. Interface constraints

2. Operating constraints

3. Economic constraints

4. Life cycle constraints: maintainability, portability, etc.

5. Performance constraints: response time, security, storage space, etc.

**Domain requirements:** in an academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement. These requirements are therefore identified from that domain model and are not user specific.

**Steps in Requirements Engineering:**

It is the process of defining, documenting and maintaining the requirements.

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software Requirement Management

**Design and Architectural Engineering**

the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.

**Characteristics of Good Design:**

1. Correctness
2. Understandability
3. Efficiency
4. Maintainability

**Modularity, Cohesion, Coupling, Layering**

**Modularization** is the process of breaking a software into multiple small modules, where each module works independently. The main advantage of modularization is that it makes it easy to understand the software, makes it reusable, and it can be tested easily.

**Coupling:** Coupling is the inter-dependency or degree of relationship between multiple modules/packages/components. Coupling is also called **Inter-module Binding.**

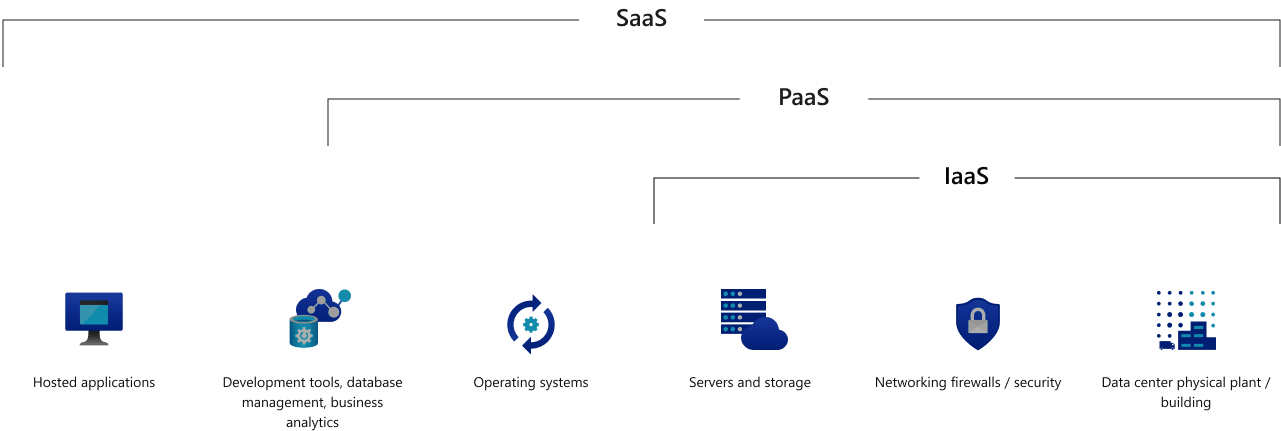
1. Data Coupling
2. Stamp Coupling
3. Control Coupling
4. External Coupling
5. Common Coupling
6. Content Coupling

**Cohesion** refers to what a module can do, internally. Cohesion should always be high means that a module/package/component is focused on what it should be doing

1. Functional Cohesion
2. Sequential Cohesion
3. Communicational Cohesion
4. Procedural Cohesion
5. Coincidental Cohesion

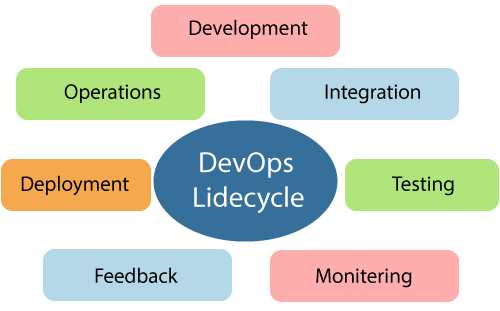
**Function Oriented vs Object Oriented System**

|  |  |  |
| --- | --- | --- |
|  | FUNCTION ORIENTED DESIGN | OBJECT ORIENTED DESIGN |
| **Abstraction** | The basic abstractions, which are given to the user, are real world functions. | The basic abstractions are not the real world functions but are the data abstraction where the real world entities are represented. |
| **Function** | Functions are grouped together by which a higher level function is obtained. | Function are grouped together on the basis of the data they operate since the classes are associated with their methods. |
| **execute** | carried out using  structured analysis and structured design i.e, data flow diagram | Carried out using UML |
| **State information** | In this approach the state information is often represented in a centralized shared memory. | In this approach the state information is not represented is not represented in a centralized memory but is implemented or distributed among the objects of the system. |
| **Approach** | It is a top down approach. | It is a bottom up approach. |
| **Begins basis** | Begins by considering the use case diagrams and the scenarios. | Begins by identifying objects and classes. |
| **Decompose** | in function/procedure level. | We decompose in class level. |
| **Use** | This approach is mainly used for computation sensitive application. | mainly used for evolving system which mimics a business or business case. |



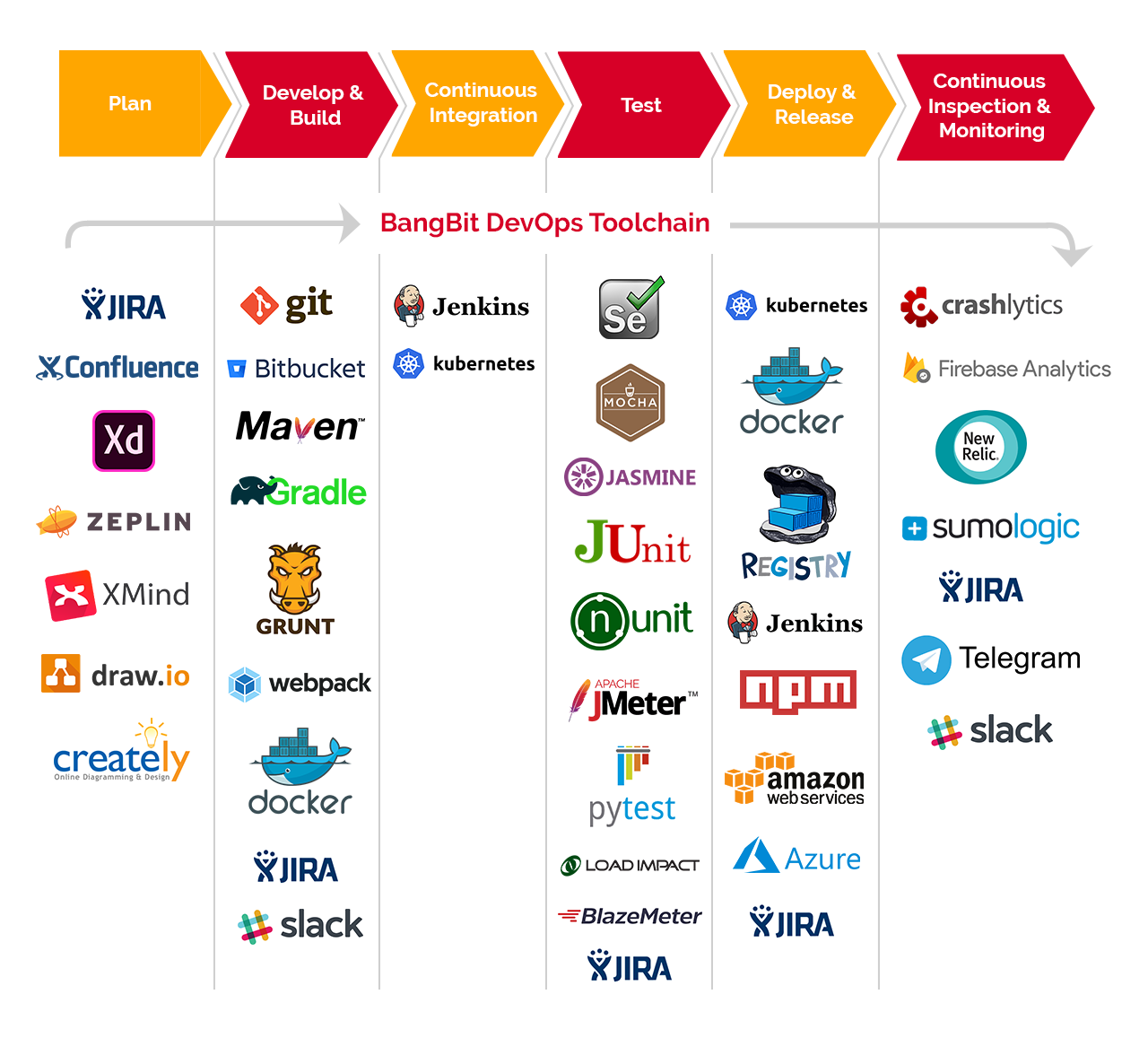
**Introduction to Microservices**

Microservice is the process of implementing Service-oriented Architecture (SOA) by dividing the entire application as a collection of interconnected services, where each service will serve only one business need. SOA is a designing pattern and Microservice is an implementation methodology to implement SOA

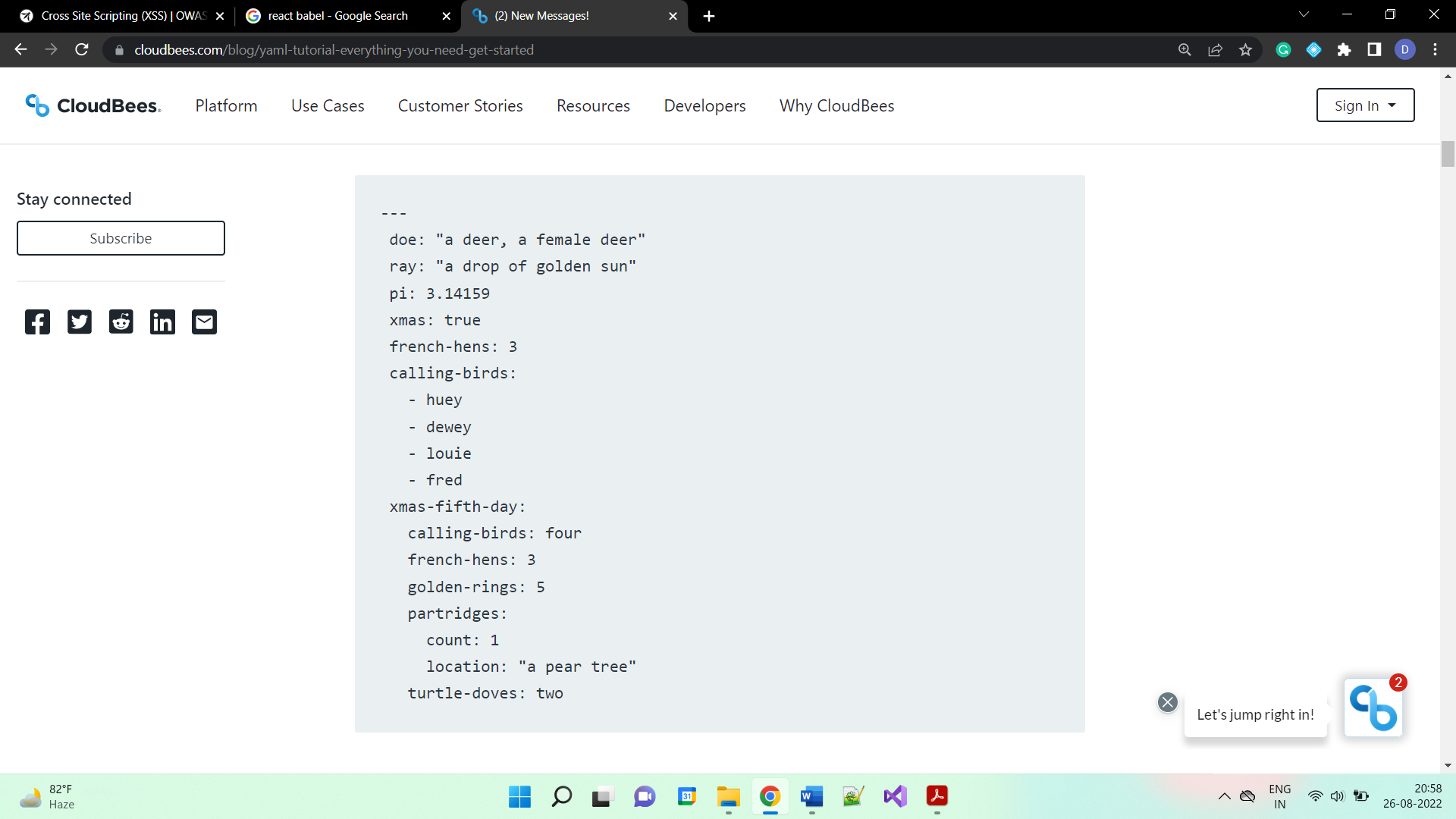


**Containerization** entails placing a software component and its environment, dependencies, and configuration, into an isolated unit called a container. This makes it possible to deploy an application consistently on any computing environment

Devops tools ecosystem.



**YAML** Ain't Markup Language (YAML) is a serialization language.

 A picture containing text

Description automatically generated

Chart

Description automatically generated with medium confidence

You enter nulls with a tilde or the unquoted null string literal.

Text

Description automatically generatedYAML strings are Unicode. In most situations, you don't have to specify them in quotes. But if we want escape sequences handled, we need to use double quotes.

YAML indicates boolean values with the keywords True, On and Yes for true. False is indicated with False, Off, or No.

**What is Software Testing**

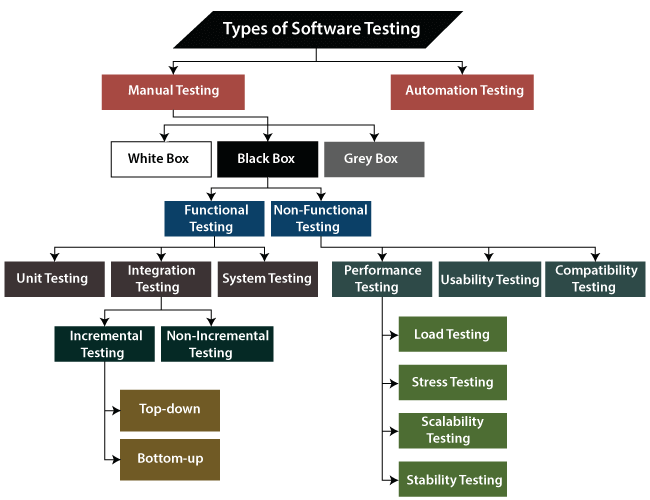
Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.

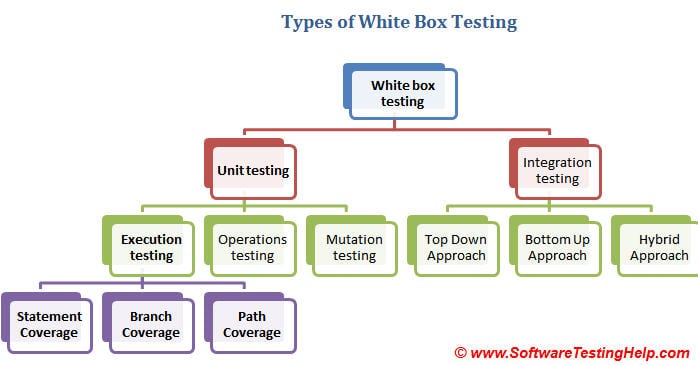
|  |  |
| --- | --- |
| **Verification** | **Validation** |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does *not* include code execution | It includes the execution of the code. |
| Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| It checks whether the software conforms to specifications or not. | It checks if the s/w meets the requirements and expectations of a customer or not. |
| It can find the bugs in the early stage of the development. | It can only find the bugs that could not be found by the verification process. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| Quality assurance team does verification. Done by humans | Validation is executed on software code with the help of testing team. Done by computer |

|  |  |  |
| --- | --- | --- |
| **Quality Assurance** | **Quality Control** | **Testing** |
| QA includes activities that ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | It includes activities that ensure the verification of a developed software with respect to documented (or not in some cases) requirements. | It includes activities that ensure the identification of bugs/error/defects in a software. |
| Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing the software with an aim to identify bug/defect through implementation of procedures and process. | Focuses on actual testing. |
| Process-oriented activities. | Product-oriented activities. | Product-oriented activities. |
| Preventive activities. | It is a corrective process. | It is a preventive process. |
| It is a subset of Software Test Life Cycle (STLC). | QC can be considered as the subset of Quality Assurance. | Testing is the subset of Quality Control. |

**Seven Principles In Software Testing:**

1. **Testing shows the presence of defects:** Software testing can ensure that defects are present, but it cannot prove that software is defect-free.
2. **Exhaustive testing is not possible**
3. **Early testing**
4. **Defect clustering:** In a project, a small number of modules can contain most of the defects. Pareto Principle to ST state that 80% of software defect comes from 20% of modules.
5. **Pesticide paradox**: Repeating the same test cases, again and again, will not find new bugs
6. **Testing is context-dependent**
7. **Absence of errors fallacy:** If a built software is 99% bug-free but it does not follow the user requirement then it is unusable.





Grey box testing is mainly used for database. Some knowledge of internal architecture. Final user view of testing. Not good for testing calculations.