**Software Development Lifecycle (SDLC)** is a systematic process to be followed for a software project. In other words, it is a structured way to create and develop software.

The main purpose of why corporations implement SDLC is to produce software with the highest quality and lowest cost in the shortest time.

There are various phases within SDLC, and each phase has its own different process and activity. This helps the development team to design, create and deliver a high-quality product.

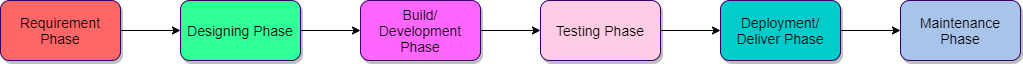
Every phase in a life cycle of software development needs to be deliverable from the previous phase. Requirements are converted into design, design into development and development into testing, after testing it is given to the client.

Software Development Lifecycle (SDLC)

Phases of SDLC

The SDLC process consists essentially of the following phases:

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



Software Development Lifecycle (SDLC)

Phase-1: Requirement Phase

The requirement is the first and the most critical phase of SDLC for both the developing team and the project manager. During this phase, the client specifies requirements, specifications, expectations and any other special requirement related to the product or software. The business manager or project manager gathers all of this information and also prerequisites.

All the information gathered from this phase is crucial to developing the product as per the customer requirements.

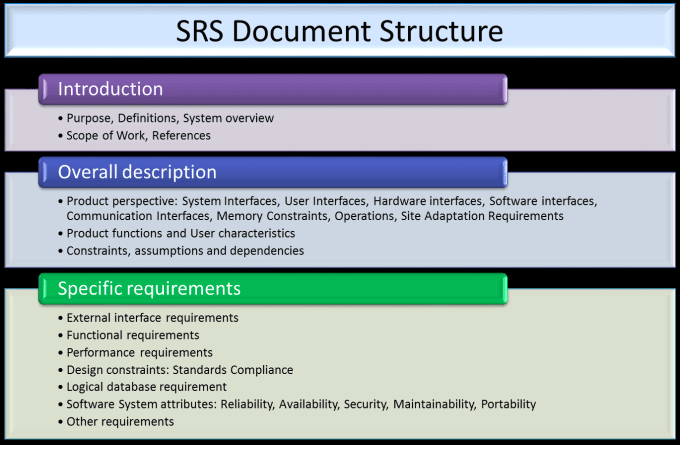
To develop the software system we should have a clear understanding of the desired product/software. To achieve this we need to continuous communication with customers to gather all requirements.

Once all the information gathered, the next step is to clearly define and document the product requirements and get them approved by the customer or the market analysts.

This is done through an **SRS (Software Requirement Specification)** document. It consists of all the necessary requirements to be designed and developed during the project life cycle.

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

**A sample of SRS Document Structure**



## Software Development Lifecycle (SDLC)

### Phase-2: Design Phase

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

This is the high priority phase in a system's development life cycle because the **logical designing** of the system is converted into **physical designing**. The output of the requirement phase is a list of things that are required and the design phase gives the way to accomplish these requirements. The decision of all required essential tools such as **programming language** like Java, .NET, PHP; **database** like Oracle, MySQL; **a combination of hardware and software** to provide a platform on which software can run without any problem is taken in this phase.

There are several tools and techniques used for describing system design, such as Flowchart, Data flow diagram (DFD), Data dictionary, Structured English, Decision table, and Decision tree.

## Software Development Lifecycle (SDLC)

### Phase-3: Build/Development Phase

After the successful completion of the requirement and design phase, the next step is to implement the design into the development of a software system.

This phase is also known as **coding phase**.

Developers start to build the entire system by writing code using the chosen programming language.

Work/task is divided into small units or modules, and coding starts by the team of developers according to the design and the requirements of the client to produce the desired result.

Coding Phase is the longest phase of the SDLC process, and it requires a more focused approach for the developer.

## Software Development Lifecycle (SDLC)

### Phase-4: Testing Phase

Once the software is complete, it is the time for the testing phase.

This phase is where you focus on investigation and discovery. The testing team starts testing the functionality of the entire system. This is done to verify that the software works and gives the result as per the requirements addressed in the requirement phase or not.

The development team makes a **test plan** to start the test. This test plan includes all types of essential testing such as integration testing, unit testing, acceptance testing, and system testing.

If there is a bug/defect detected in the software, or it is not working as expected. The testing team gives detailed information to the development team about the issue. If the defect is valid or worth fixing, it will be fixed and the development team replaces it with the new one. It also needs to be verified.

## Software Development Lifecycle (SDLC)

### Phase-5: Deployment/Deliver Phase

When software testing is completed with a satisfying result and there are no remaining issues in the working of the software, it is delivered to the customer.

As soon as customers receive the product, they are recommended first to do the beta testing. In beta testing, customers can require any changes which are not present in the software but mentioned in the requirement document to make it more user-friendly.

Besides this, if any type of defect is encountered while a customer using the software, the development team will be informed to fix this problem. If it is a critical defect, the development team solves it in a short time. Otherwise, it will wait for the next version.

After the solution of all types of bugs and changes, the software finally deployed to the end-user.

Software Development Lifecycle (SDLC)

Phase-6: Maintenance Phase

The **last** phase of the process SDLC is the maintenance phase where the process continues until the software's life cycle comes to an end. When a customer starts using software, actual problems start to show up. At that time, there's a need to solve these problems.

Maintenance Phase also includes making changes in hardware and software to maintain its operational effectiveness like to improve its performance, enhance security features and address customer's requirements.Formun Üstü

## SDLC Models

### Introduction

A SDLC model describes the types of activities performed in a software development project at each stage, and how the activities relate logically and chronologically to each other.

There are many different SDLC models, each of which requires different approaches to testing.

## SDLC Models

### Software Development and Software Testing

It is an important part of a tester's role to be familiar with the common SDLC models so that appropriate test activities can take place. In any SDLC model, there are several characteristics of good testing:

* For every development activity, there is a corresponding test activity
* Each test level has test objectives specific to that level
* Test analysis and design for a given test level begin during the corresponding development activity
* Testers participate in discussions to define and refine requirements and design, and they are involved in reviewing work products as soon as drafts are available

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

## SDLC Models

### Verification & Validation

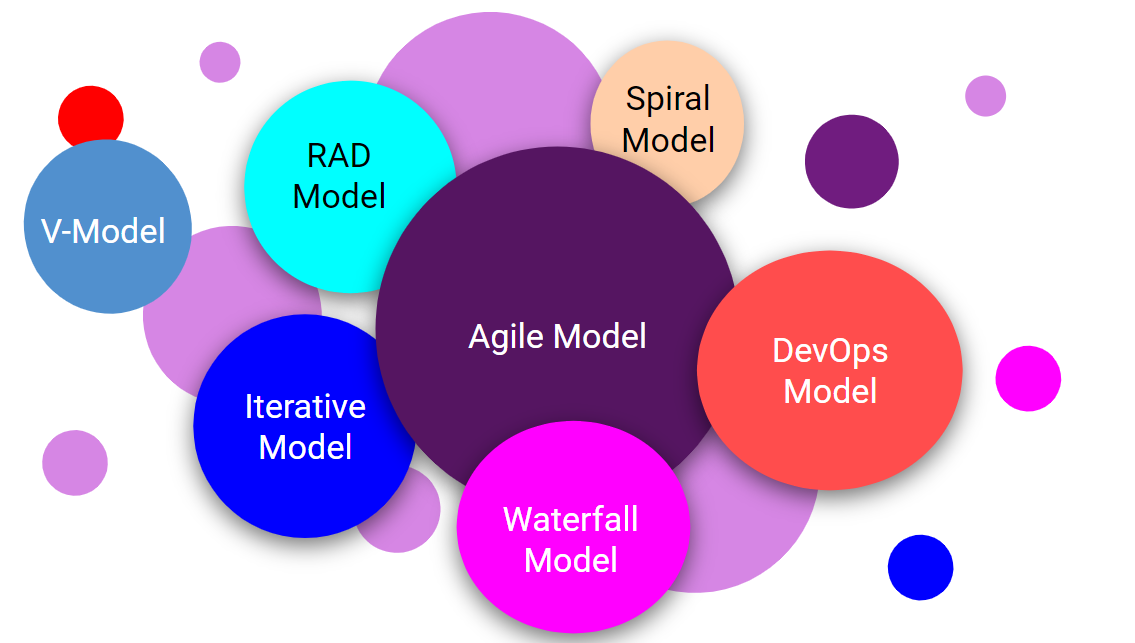
In every development life cycle, a part of testing is focused on **verification** testing, and another part is focused on **validation** testing.

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

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

### SDLC Models

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



## Waterfall Model

### Waterfall Model

**The Waterfall Model** is the earliest approach and a widely accepted SDLC model that was used for software development to ensure the success of some type of projects.

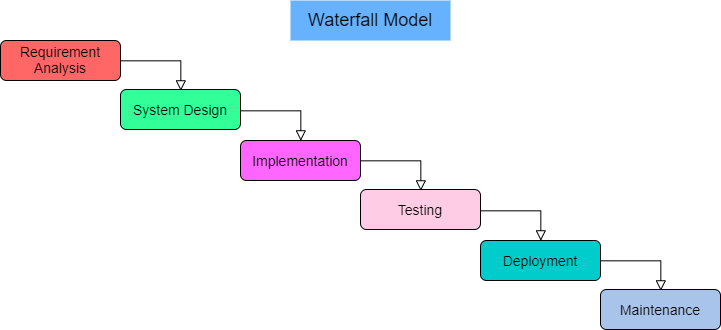
This model is very simple to understand and use. The model formed the transition between the phases of the software development process like a waterfall pouring.

It illustrates the process of software development in a **linear sequential flow**. This means that each phase must be completed before the beginning of the next phase. Each phase finds what the previous phase produces at the starting point and there is no overlapping in the phases.

## Waterfall Model

### Design

In **The Waterfall** approach, the entire software development process is divided into separate phases. Usually, In this model, the result of one phase acts sequentially as the input for the next phase.



The sequential phases in the Waterfall model are;

* **Requirement Gathering and analysis** − During this process, all possible system requirements to be created are collected and recorded in a requirement specification document.
* **System Design** − In this phase, the requirements from the first phase will be reviewed, and the device design will be prepared. This system design helps determine the specifications of the hardware and system and helps define the overall system architecture.
* **Implementation** − The system is first developed with inputs from the system design into small programs called units, which are integrated into the next phase. Each unit is developed and tested for their functionality, known as Unit Testing.
* **Integration and Testing** − After testing each unit, all the units built during the implementation process are incorporated into a system. Any flaws and deficiencies are checked for the post-integration of the entire system.
* **Deployment of system** − Once testing is done, the product is deployed in the customer environment or released into the market.
* **Maintenance** − Some issues arise in the client environment. Patches are released to fix those issues. Some improved versions are released to enhance the product. Maintenance is done in the consumer environment to make such improvements.

## Waterfall Model

### Application

Each software developed is different and requires an appropriate SDLC approach based on both internal and external factors to be followed. Many situations where the most effective use of the Waterfall model are;

* The requirements are documented very well.
* The definition of a product is stable.
* Technology is comprehensible.
* Requirements are not ambiguous.
* Ample resources are available with the required expertise to support the product.
* The project is short.

## Waterfall Model

### Advantages

Some of the major advantages of the Waterfall Model are as follows;

* It allows control and departmentalization.
* A schedule may be set with deadlines for each stage of development, and a product may proceed one by one through phases of the development process model.
* Every development phase proceeds in strict order.
* Easy and simple to use and understand.
* Easy to manage because of model rigidity. There are specific deliverables and a review process for each phase.
* Phases are processed one at a time and completed.
* Works well for smaller projects where requirements are very well understood.
* Phases clearly defined.
* The coding and testing steps are very short, as the requirements and design are clearly defined during the analysis and design phases.
* The number of errors during the test phase is very small.
* Tasks are easy to arrange.
* There is good documentation of the process and results.

## Waterfall Model

### Disadvantages

Some of the major disadvantages of the Waterfall Model are as follows;

* It does not allow a great deal of reflection or revision.
* Once an application is in the phase of testing, it is very hard to go back and alter something that was not well-documented.
* No working software is produced until late during the life cycle.
* There is a high amount of risk and uncertainty.
* The time-loss due to the upper phase errors is quite high.
* It is not a good model for complex and object-oriented projects.
* It is a poor model for long and ongoing projects.
* It is not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty are high with this process model.
* The progress in phases can be difficult to measure.
* The product has to wait until the end of all the phases.
* It can not adapt to changing requirements.
* Scope adjustment over the life cycle can end a project.

## Iterative Model (Optional)

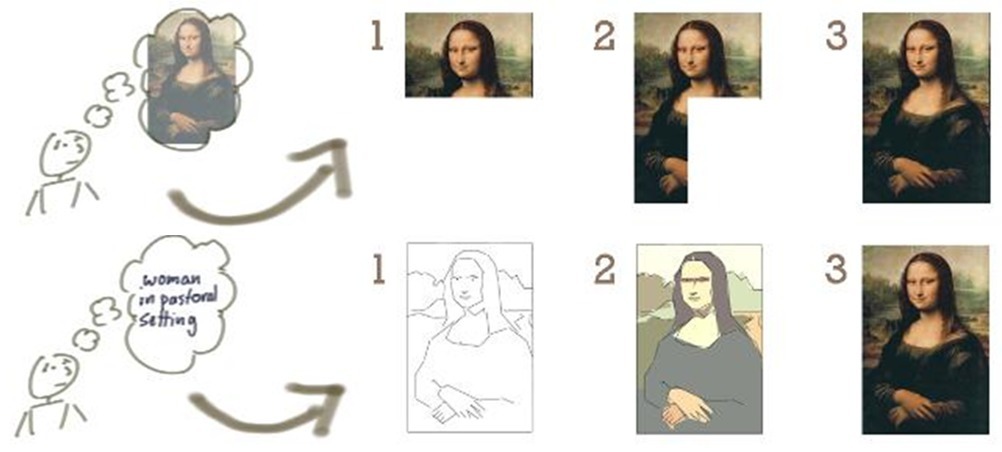
### Iterative Model

In the Iterative Model, the process starts with a simple implementation of a subset of the software requirements. At each iteration, design modifications are made and new functional capabilities are added. At the end of every iteration, a version of the software is produced.

Errors, deficiencies, and disturbances that occur later are resolved. The errors caused by the previous step are removed.

Iterations may involve changes to features developed in earlier iterations, as well as changes in project scope.

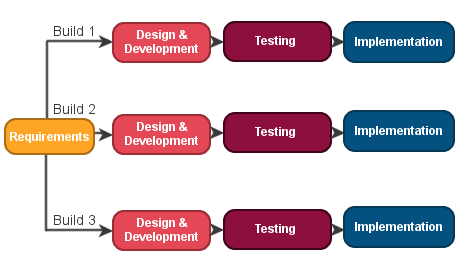
Every iteration delivers working software until the final software is delivered or development stops.



## Iterative Model (Optional)

### Design

The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).



In the incremental model, the entire requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation, and testing phases. Each subsequent release of the module adds function to the previous release. The process continues until the complete system is ready as per the requirement.

The key to the successful use of an iterative life cycle of software development is to rigorously validate requirements and test each version of the software against those requirements within each model cycle.

Tests must be repeated and extended to verify each version of the software as the software evolves through successive cycles.

## Iterative Model (Optional)

### Application

Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios;

* The requirements of the entire system are clearly defined and understood.
* Specific requirements need to be defined; however, some functionalities or required improvements may evolve with time.
* There is a time for market constraint.
* The development team is using new technology and learning while working on the project.
* Some certain high-risk features and goals may change in the future.

## Iterative Model (Optional)

### Advantages

The advantages of the Iterative and Incremental SDLC Model are as follows;

* It allows the team to find functional or design related flaws as early as possible.
* Parallel development can be planned.
* It is best suited for agile organizations.
* Implemented during the earlier stages of the development process.
* It supports changing requirements.
* It spends less time on documenting and more time on designing.

## Iterative Model (Optional)

### Disadvantages

The disadvantages of the Iterative and Incremental SDLC Model are as follows;

* While change costs are lower, they are not very suitable for changing requirements.
* It may need more management attention.
* It is not suitable for smaller projects.
* Defining increments may require a complete system definition.
* Project success and progress depend heavily upon the process of risk analysis.
* To implement iterative model more resources may be required.

## Spiral Model (Optional)

### Spiral Model

The spiral model is a **risk-driven model** of software process development.

The spiral model, based on a given project's unique risk patterns, guides a team to follow elements of one or more process models, such as incremental or waterfall.

This model looks like a spiral with a lot of loops in its diagrammatic representation. The exact number of spiral loops is uncertain and may vary from project to project. Every spiral loop is called a phase of the software development process. Depending on the project risks, the exact number of phases required to develop the product can be varied by the project manager.

The project manager has an important role to play in developing a product using the spiral model because the project manager determines the number of phases dynamically.

## Spiral Model (Optional)

### Design

Each phase of the Spiral Model is broken down into four phases.



**Objectives determination and identify alternative solutions**

* Requirements are obtained from the clients.
* At the beginning of every phase, the goals are defined, elaborated and analyzed.
* Then alternative solutions possible for the phase are proposed.

**Identify and resolve Risks**

* All the possible solutions are evaluated to select the best possible solution.
* The risks associated with that solution are identified.
* The risks are resolved using the best possible strategy.
* At the end of this quadrant, Prototype is built for the best possible solution.

**Develop next version of the Product**

* The identified features are developed and verified through testing.
* At the end of this quadrant, the next version of the software is available.

**Review and plan for the next Phase**

* Customers evaluate **the so far developed** version of the software.
* In the end, planning for the next phase is started.

## Spiral Model (Optional)

### Application

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product.

The typical uses of a Spiral Model are as follows;

* When there is a constraint on the budget and the risk evaluation is important.
* It is appropriate for projects which are of medium to high risk.
* When there is a long-term project commitment as the criteria change with time due to potential changes to economic goals.
* It is appropriate when the customer is not sure what is usually the case for their requirements.
* Requirements are complex and need evaluation to get clarity.
* The new product line which should be released in phases to get enough customer feedback.
* Significant changes are expected in the product during the development cycle.

## Spiral Model (Optional)

### Advantages

The advantages of the Spiral SDLC Model are as follows;

* **It is one of the best development models to follow because of the risk analysis and risk handling at every phase.**
* This model can help to change requirements easily.
* It allows extensive use of prototypes.
* Requirements can be more reliably and accurately identified.
* It allows users to see the system early.
* Development can be divided into smaller components, and the risky components can be developed earlier, which helps to improve risk management.

## Spiral Model (Optional)

### Disadvantages

The disadvantages of the Spiral SDLC Model are as follows;

* It is much more complex to manage than other SDLC models.
* As the number of phases is unknown at the start of the project, so time estimation is very difficult.
* It is not suitable for small or low-risk projects
* It could be expensive for small projects.
* Spiral can go on indefinitely.
* Due to the excess of intermediate phases, it requires a lot of documentation.

## V-Model (Optional)

### V-Model

The V-model is a SDLC model where the execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model.**

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage.

The next phase starts only after the completion of the previous phase.

## V-Model (Optional)

### Design

The corresponding testing phase of the development phase is planned in parallel.

There are **Verification** phases on one side of the ‘V’ and **Validation** phases on the other side.

The V-Model's two sides are joined by the Coding Phase.



## V-Model (Optional)

### Verification Phases

Verification phases in the V-Model are as follows;

* **Requirement Analysis** -
  + This phase is also known as the **Requirement Gathering** phase.
  + This is the first phase that involves detailed communication with the customer to understand their requirements and expectations.
* **System Design** -
  + This phase contains the system design and the complete hardware and communication setup.
* **Architectural Design** -
  + Architectural specifications are understood and designed in this phase.
  + The system design is further broken down into modules that take on various functionalities. This is also referred to as **High-Level Design (HLD)**.
  + There is a clear understanding of the data transfer and connectivity between the internal modules and the outside world (other systems).
* **Module Design** -
  + In this phase, the system breaks down into small modules.
  + The detailed design of modules is specified, also known as **Low-Level Design (LLD)**.
  + The design is compatible with the other modules in the system architecture and the other external systems.
* **Coding Phase** -
  + In this phase, the actual coding of the system modules designed in the design phase is taken up.
  + The best suitable programming language is decided based on the system and architectural requirements.
  + The coding is done according to the coding guidelines and standards.
  + The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

## V-Model (Optional)

### Validation Phases

Validation phases in the V-Model are as follows;

* **Unit Testing** -
  + The unit test is designed in the module design phase.
  + It is executed to eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.
* **Integration Testing** -
  + Integration testing is associated with the architectural design phase.
  + Integration tests are performed to test the coexistence and communication of the internal modules within the system.
* **System Testing** -
  + System testing is directly associated with the system design phase.
  + System tests check the entire functionality of the system and the communication of the system under development with external systems.
  + During this system test execution, most software and hardware compatibility issues can be uncovered.
* **Acceptance Testing** -
  + Acceptance testing is associated with the requirement analysis phase.
  + It involves testing the product in the user environment.
  + It uncovers the compatibility issues with the other systems available in the user environment.
  + It also finds the non-functional issues in the actual user environment, such as load and performance defects.

## V-Model (Optional)

### Application

This model is used in the medical development field.

The typical uses of a V-Model are as follows;

* The requirements are documented clearly.
* The requirements are well defined.
* Requirements are not ambiguous.
* The definition of a product is stable.
* Technology is not dynamic.
* It is well understood by the project team.
* The project is short.

## V-Model (Optional)

### Advantages

The advantages of the V-Model method are as follows;

* It is very easy and simple to understand and apply.
* It is easy to manage due to the rigidity of the model.
* It is a highly-disciplined model.
* Phases are completed one at a time.
* It works well for smaller projects where requirements are very well understood.
* Every phase has specific deliverables and a review process.

## V-Model (Optional)

### Disadvantages

The disadvantages of the V-Model method are as follows;

* It is not flexible to changes.
* There is a high amount of risk and uncertainty.
* It is not a good model for complex and object-oriented projects.
* It is a poor model for long and ongoing projects.
* It is not suitable for the projects where requirements are at a moderate to high risk of changing.
* When an application is in the testing stage, it is difficult to go back and change functionality.
* No working software is produced until late during the life cycle.

## RAD Model (Optional)

### RAD Model

The **RAD (Rapid Application Development)** is a model that supports software development processes to be carried out as quickly as possible.

The RAD model focuses on iterative and incremental delivery of working models to the customer.

This model is based on prototyping and results in rapid delivery.

For the users to adapt to the system quickly, prototypes that give an idea and revive the system are made.

This model uses minimal planning in favor of rapid prototyping. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.

The main objectives of this method are; can be explained as high speed, high quality, and low cost.

The purpose of this model is not to reveal fast and scattered software. On the contrary, this model aims to speed up the process by drawing attention to important points and increase the quality while accelerating the process with these points.

The end-users should be included in the software design and ask them to indicate their requests to the finest detail through workshops or focus groups.

Customer involvement reduces the risk of non-conformance with the actual user requirements during the complete development cycle of the product.

The most important aspect of this model to be successful is to make sure that the prototypes developed are reusable.

## RAD Model (Optional)

### Design

RAD model distributes the analysis, design, build and test phases into a series of short, iterative development cycles.



Following are the various phases of the RAD Model;

**Business Modeling** - The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. Complete business analysis is performed to find vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving the successful flow of information.

**Data Modeling** - The information gathered in the Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets are identified and defined. The relation between these data objects is established and defined in detail in relevance to the business model.

**Process Modeling** - The data object sets defined in the Data Modeling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding, deleting, retrieving or modifying a data object are given.

**Application Generation** - The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

**Testing and Turnover** - The overall testing time is reduced in the RAD model as the prototypes are independently tested during every iteration. However, the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

## RAD Model (Optional)

### Application

RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail.

The following pointers describe the typical scenarios where RAD can be used;

* RAD should be used only when a system can be modularized to be delivered in an incremental manner.
* It should be used if there is a high availability of designers for modeling.
* It should be used only if the budget permits the use of automated code generating tools.
* RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
* It should be used where the requirements change during the project and working prototypes are to be presented to the customer in small iterations of 2-3 months.

## RAD Model (Optional)

### Advantages

The advantages of the RAD Model are as follows;

* Changing requirements can be accommodated.
* Progress can be measured.
* Iteration time can be short with the use of powerful RAD tools.
* Productivity with fewer people in a short time.
* Reduced development time.
* Increases the reusability of components.
* Quick initial reviews occur.
* Encourages customer feedback.
* Integration from the very beginning solves a lot of integration issues.

## RAD Model (Optional)

### Disadvantages

The disadvantages of the RAD Model are as follows;

* Dependency on technically strong team members for identifying business requirements.
* The only system that can be modularized can be built using RAD.
* Requires highly skilled developers/designers.
* High dependency on modeling skills.
* Inapplicable to cheaper projects as the cost of modeling and automated code generation is very high.
* Management complexity is more.
* Suitable for systems that are component-based and scalable.
* Requires user involvement throughout the life cycle.
* Suitable for project requiring shorter development times.

## DevOps Model

### What is DevOps

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

It focuses on collaboration between developers and other roles.

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

DevOps aims to shorten the system’s development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.

DevOps is an evolution of the Agile Model of software development.

As the Agile model addressed the gap between clients and developers, DevOps addressed the gap between Developers and Operations.

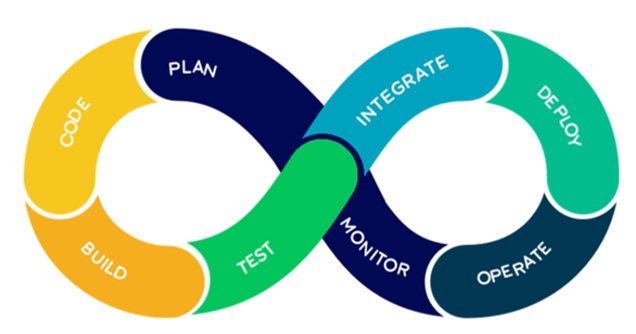
The development team will submit the application to the operations team for implementation.

The operations team will monitor the application and provide relevant feedback to developers.

## DevOps Model

### DevOps Phases

According to DevOps practices, the workflow in software development and delivery is divided into 8 phases.



**Plan** - Business owners and software development team discuss project goals and create a plan.

**Code** - Programmers then design and code the application and use tools like Git to store application code.

**Build** - Build tools like Maven and Gradle, take code from different repositories and combine them to build the complete application.

**Test** - Application is tested using automation testing tools like Selenium and Junit to ensure software quality.

**Integrate** - When testing is complete, new features are integrated automatically to the already existing codebase.

**Deploy** - Application is packaged after release and deployed from the development server to the production server.

**Operate** - Once the software is deployed, the operations team performs activities such as configuring servers and provisioning them with the required resources.

**Monitor** - Monitoring allows IT organizations to identify specific issues of specific releases and understand the impact on end-users.

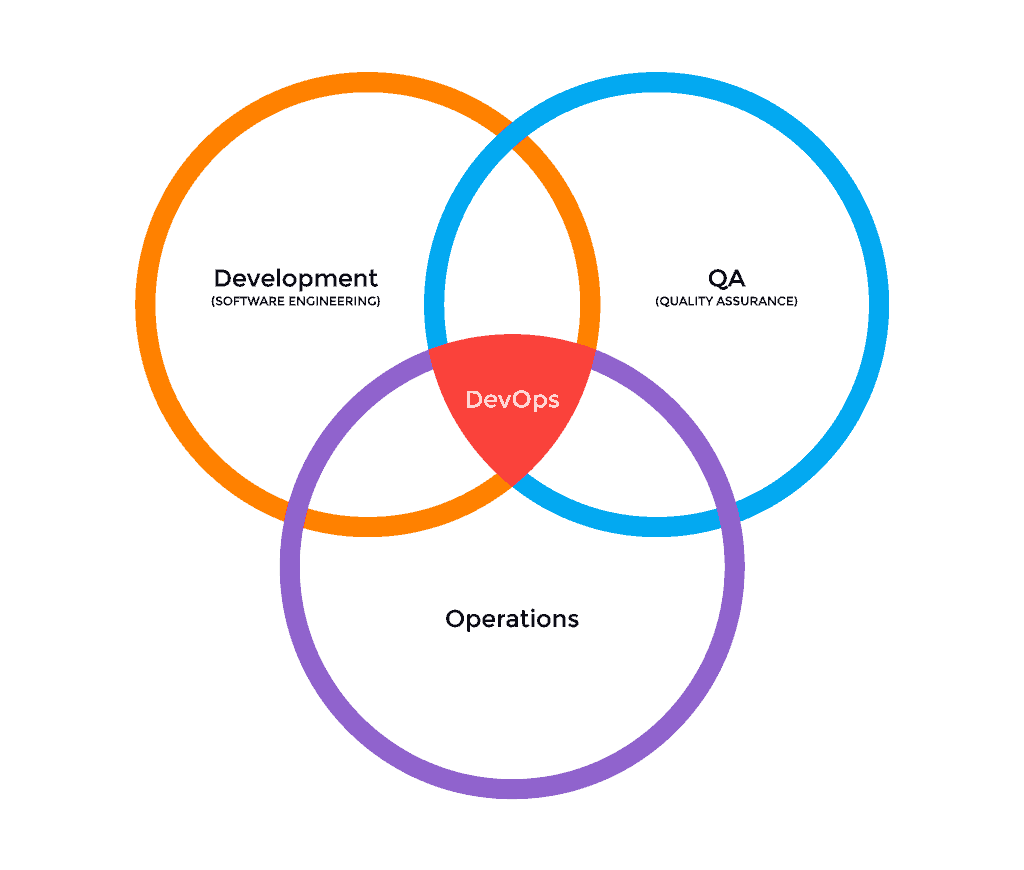
## DevOps Model

### DevOps Model

The DevOps is a new SDLC model that focuses on communication, collaboration, integration between Developers and Operations teams to accelerate innovation and the deployment of higher-quality and more reliable software products.

DevOps is a culture that promotes collaboration between Development and Operations teams. This allows deploying code to production faster and in an automated way. It helps to increases an organization's speed to deliver applications and services.

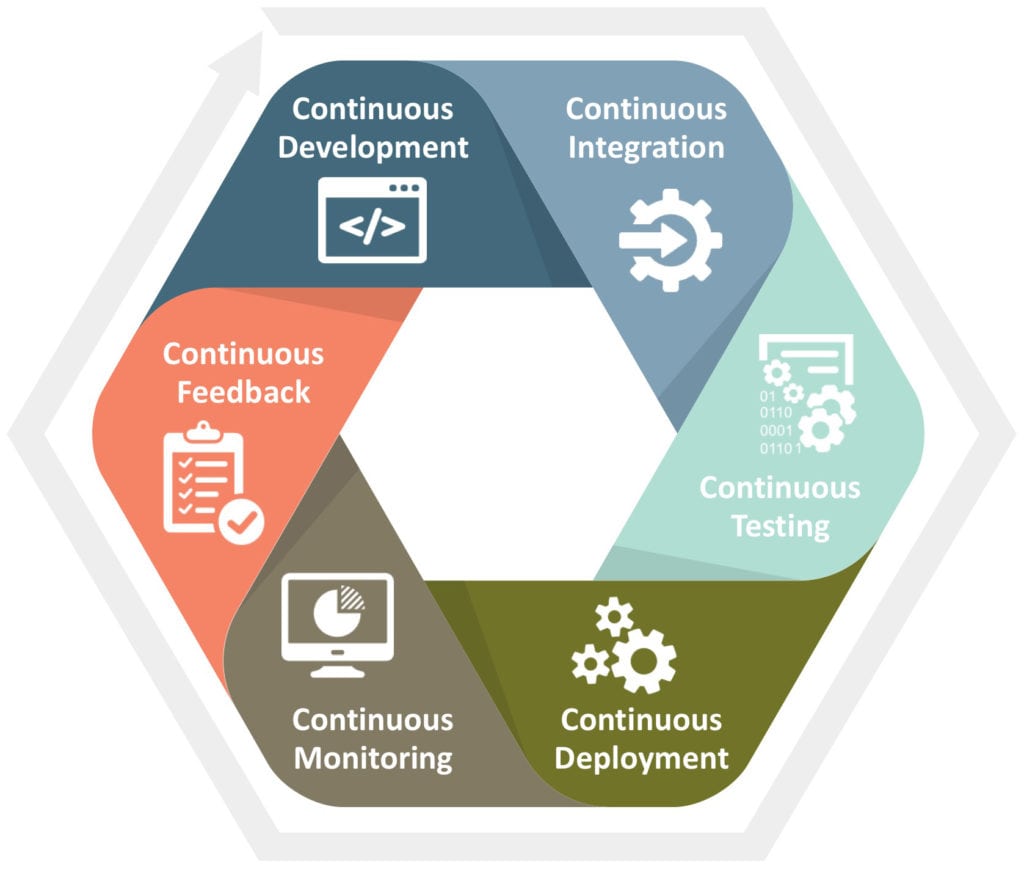
DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.



## DevOps Model

### DevOps Processes

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



**Continuous Development**

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

**Continuous Testing**

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

**Continuous Integration**

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

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

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

**Jenkins** is a very popular tool used in this phase.

**Continuous Deployment**

This is the stage where the code is deployed to the production servers.

It is also important to ensure that the code is correctly deployed on all the servers.

**Continuous Monitoring**

This is a very crucial stage of the DevOps life cycle, where it is continuously monitored the performance of the application.

Here vital information about the use of the software is recorded.

This information is processed to recognize the proper functionality of the application.

The system errors such as low memory, server not reachable, etc. are resolved in this phase.

The root cause of any issue is determined in this phase.

It maintains the security and availability of the services.

## DevOps Model

### DevOps Principles

Some of the DevOps principles are;

* Create production-like systems for development and testing environment
* Deployments need to iterative and frequent. Ensure a reliable and repeatable process
* Continuously monitor and validate operational quality characteristics
* Amplify feedback loop

## DevOps Model

### Advantages

Some of the DevOps advantages are;

* Time taken to create and deliver software is reduced
* The complexity of maintaining an application is reduced
* Improved collaboration between developers and operations team
* Continuous integration and delivery ensure faster time to market

## DevOps Model

### Disadvantages

Some of the DevOps disadvantages are;

* Technology investments in the automation tools required for DevOps are **costly** and will take a great deal of time to identify and implement.
* It needs a **specialist** who can cover each stage of the software delivery pipeline rather than investing in a smaller number of full-stack developers.
* It is easy to add new features, but more features may not always better, even if they can be implemented efficiently.

Formun Altı