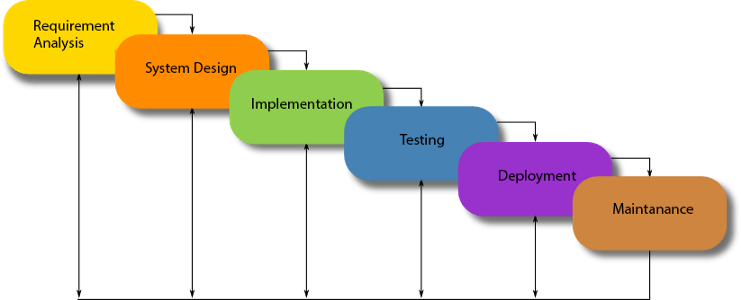
**Module - 1 (Fundamental)**

**What is SDLC**

SDLC is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase. SDLC stands for Software Development Life Cycle and is also referred to as the Application Development life-cycle.

**SDLC Phases**

The entire SDLC process divided into the following SDLC steps:



Phase 1: Requirement collection and analysis

Phase 2: Feasibility study

Phase 3: Design

Phase 4: Coding

Phase 5: Testing

Phase 6: Installation/Deployment

Phase 7: Maintenance

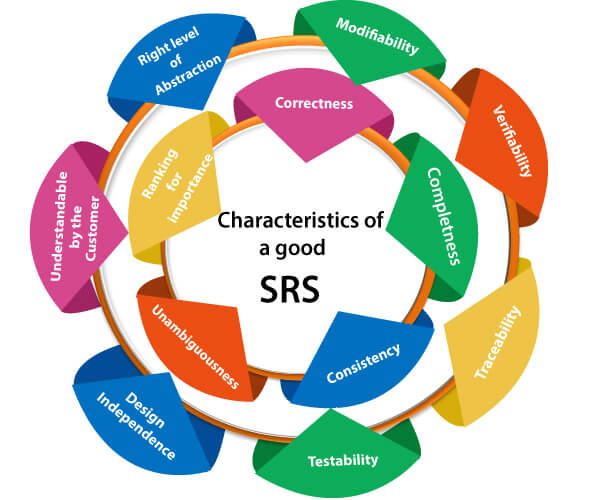
**What is agile methodology?**

Agile Methodology meaning a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.

**What is SRS**

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS) (also called a requirements document). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. SRS is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The two methods create entirely various situations and establish different purposes for the document altogether. The first case, SRS, is used to define the needs and expectation of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.



**What is oops**

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained. This includes programs for manufacturing and design, as well as mobile applications; for example, OOP can be used for manufacturing system simulation software.

The organization of an object-oriented program also makes the method beneficial to collaborative development, where projects are divided into groups. Additional benefits of OOP include code reusability, scalability and efficiency.

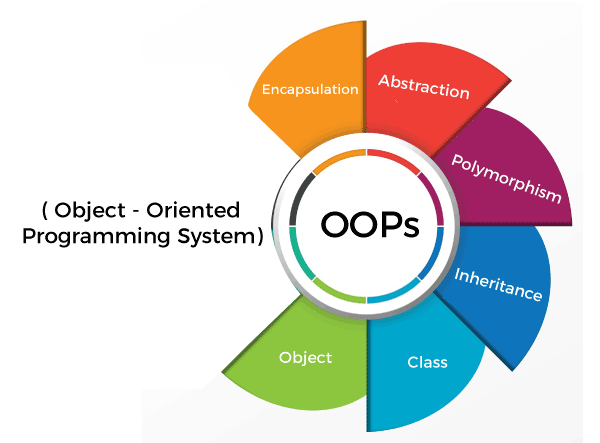
The first step in OOP is to collect all of the objects a programmer wants to manipulate and identify how they relate to each other -- an exercise known as data modeling.

Examples of an object can range from physical entities, such as a human being who is described by properties like name and address, to small computer programs, such as widgets.

Once an object is known, it is labeled with a class of objects that defines the kind of data it contains and any logic sequences that can manipulate it. Each distinct logic sequence is known as a method. Objects can communicate with well-defined interfaces called messages.

**Write Basic Concepts of oops**

Object oriented programming is a type of programming which uses objects and classes its functioning. The object oriented programming is based on real world entities like inheritance, polymorphism, data hiding, etc. It aims at binding together data and function work on these data sets into a single entity to restrict their usage.



**Some basic concepts of object oriented programming are −**

1. CLASS
2. OBJECTS
3. ENCAPSULATION
4. POLYMORPHISM
5. INHERITANCE
6. ABSTRACTION

**Class** − A class is a data-type that has its own members i.e. data members and member functions. It is the blueprint for an object in object oriented programming language. It is the basic building block of object oriented programming in c++. The members of a class are accessed in programming language by creating an instance of the class.

**Some important properties of class are −**

* Class is a user-defined data-type.
* A class contains members like data members and member functions.
* Data members are variables of the class.
* Member functions are the methods that are used to manipulate data members.
* Data members define the properties of the class whereas the member functions define the behaviour of the class.

A class can have multiple objects which have properties and behaviour that in common for all of them.

**Object −** An object is an instance of a class. It is an entity with characteristics and behaviour that are used in the object oriented programming. An object is the entity that is created to allocate memory. A class when defined does not have memory chunk itself which will be allocated as soon as objects are created.

**Encapsulation-** In object oriented programming, encapsulation is the concept of wrapping together of data and information in a single unit. A formale defination of encapsulation would be: encapsulation is binding togather the data and related function that can manipulate the data.

**Polymorphism-** The name defines polymorphism is multiple forms. which means polymorphism is the ability of object oriented programming to do some work using multiple forms. The behaviour of the method is dependent on the type or the situation in which the method is called.

**Inheritance-** Inheritance it is the capability of a class to inherit or derive properties or characteristics other class. it is very important and object oriented program as it allows reusability i.e. using a method defined in another class by using inheritance. The class that derives properties from other class is known as child class or subclass and the class from which the properties are inherited is base class or parent class.

C plus plus programming language supports the following types of inheritance

1. single inheritance
2. multiple inheritance
3. multi level inheritance
4. Hierarchical inheritance
5. hybrid inheritance

**Abstraction-** Abstraction Data abstraction or Data Hiding is the concept of hiding data and showing only relevant data to the final user. It is also an important part object oriented programing.

**What is object**

**Object**



Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

**Example:** A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

**What is class**

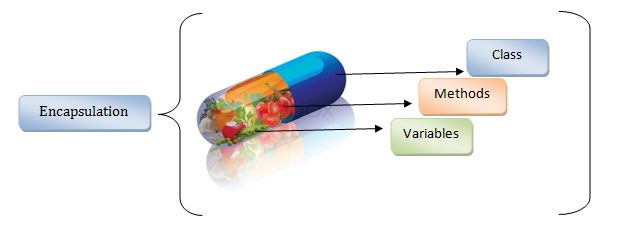
**Class**

Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

**What is encapsulation**

**Encapsulation**



Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

**What is inheritance**

**Inheritance**

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**What is polymorphism**

**Polymorphism**



If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

**What is RDBMS**

A relational database management system (RDBMS) is a collection of programs and capabilities that enable IT teams and others to create, update, administer and otherwise interact with a relational database. RDBMSes store data in the form of tables, with most commercial relational database management systems using Structured Query Language (SQL) to access the database. However, since SQL was invented after the initial development of the relational model, it is not necessary for RDBMS use.

The RDBMS is the most popular database system among organizations across the world. It provides a dependable method of storing and retrieving large amounts of data while offering a combination of system performance and ease of implementation.

**What is SQL**

SQL stands for Structured Query Language. It is a programming language that is used to request information from a database. SQL can be used to manage and share data in a relational database management system. Moreover, users can perform actions like insertion, deletion, selection, etc on the database.

SQL programming is widely used as using this language we can create, store as well as manipulate the data inside the database. Moreover, SQL can be embedded within other languages through SQL libraries and modules.

**Write SQL Commands**

Some of The Most Important SQL Commands

* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database
* CREATE DATABASE - creates a new database
* ALTER DATABASE - modifies a database
* CREATE TABLE - creates a new table
* ALTER TABLE - modifies a table
* DROP TABLE - deletes a table
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index

**Draw Usecase on Online book shopping**

Select category

Log in

Search book

<<extend>>

Advanced search

<<extend>>

Place order

<<uses>>

Add item to cart

Payment

<<uses>>

Validate card details

Registration

Update profile

**Draw Usecase on online bill payment system (paytm)**

**Accounts**

**Cashback**

**Tools and investment**

**Bill pay**

**Transfer**

**Open account**

<<include>>

**Paytm/online banking**

**Existing User**

**Login using username/ password**

**Verify username/ password**

**Bill pay centre**

**Ebill**

**Payment activity**

**Automatic payment**

**Manage pay to**

**Bank Server**

**Add a new pay to account**

**Display user account dashboard**

**Enter amount**

**Check balance**

<<include>>

**Select date**

**Make payment**

**Send email**

<<extend>>

**.**

**Write SDLC phases with basic introduction?**



**#1) Requirement Gathering and Analysis**

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.

**#2) Design**

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.

**#3) Implementation or Coding**

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

**#4) Testing**

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

Retesting, regression testing is done until the point at which the software is as per the customer’s expectation. Testers refer SRS document to make sure that the software is as per the customer’s standard.

**#5) Deployment**

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation.

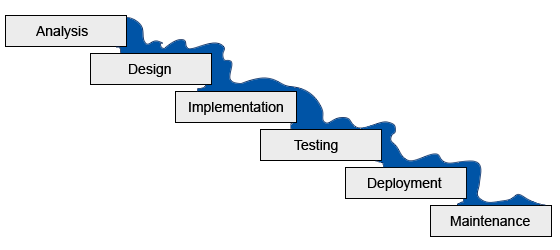
In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

**#6) Maintenance**

After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

**Explain Phases of the waterfall model**

In the waterfall model, the individual phases of a development process are arranged in a cascade. Each phase concludes with an intermediate result (milestone) – for example with a catalogue of requirements in the form of a requirement specification, with the specification of a software architecture or with an application in the alpha or beta stage.



**Analysis**

Every software project begins with an analysis phase that includes a feasibility study and a requirements definition. In the feasibility study, the software project is assessed in terms of costs, revenue, and feasibility. The feasibility study provides a requirement specification (a rough description of the requirements), a project plan and the project calculation, as well as an offer for the client, if applicable.

This is followed by a detailed definition of the requirements, which includes an analysis of the current situation and a target concept. While as-is analyses outline the problem area, the target concept defines which functions and properties the software product must offer in order to meet the requirements. The results of the requirements’ definition include, for example, a requirement specification, a detailed description of how the project requirements are to be met, and a plan for acceptance testing.

Finally, the first phase of the waterfall model provides for an analysis of the requirements’ definition, in which complex problems are broken down into small subtasks and appropriate solution strategies are developed.

**Design**

The design phase serves to develop a concrete solution concept based on the previously determined requirements, tasks, and strategies. In this phase, software developers develop the software architecture and a detailed construction plan for the software, concentrating on specific components such as interfaces, frameworks, or libraries. The result of the design phase comprises a draft document with a software construction plan and test plans for individual components.

**Implementation**

The software architecture designed in the design phase is implemented in the implementation phase, which includes software programming, troubleshooting, and module testing. In the implementation phase, the software design is implemented in the desired programming language. Individual components are developed separately, checked within the framework of module testing, and integrated step by step into the overall product. The result of the implementation phase is a software product that is tested for the first time as a complete product in the subsequent phase (alpha test).

**Testing**

The test phase includes the integration of the software into the desired target environment. As a rule, software products are initially delivered as beta versions to selected end users (beta tests). The acceptance tests developed in the analysis phase can be used to determine whether the software meets the previously-defined requirements. A software product that has successfully completed beta testing is ready for release.

**Maintenance**

After successful completion of the test phase, the software is released for productive use. The final phase of the waterfall model includes delivery, maintenance, and improvement of the software.

**Write phases of spiral model**

It has four stages or phases: The planning of objectives, risk analysis, engineering or development, and finally review. A project passes through all these stages repeatedly and the phases are known as a Spiral in the model.

1. **Determine objectives and find alternate solutions** – This phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.
2. **Risk Analysis and resolving** – In this quadrant, all the proposed solutions are analyzed and any potential risk is identified, analyzed, and resolved.
3. **Develop and test**: This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.
4. **Review and planning of the next phase** – In this phase, the software is evaluated by the customer. It also includes risk identification and monitoring like cost overrun or schedule slippage and after that planning of the next phase is started.

**Write agile manifesto principles**

The 12 principles articulated in the Agile Manifesto are:

1. Satisfying customers through early and continuous delivery of valuable work.
2. Breaking big work down into smaller tasks that can be completed quickly.
3. Recognizing that the best work emerges from self-organized teams.
4. Providing motivated individuals with the environment and support they need and trusting them to get the job done.
5. Creating processes that promote sustainable efforts.
6. Maintaining a constant pace for completed work.
7. Welcoming changing requirements, even late in a project.
8. Assembling the project team and business owners on a daily basis throughout the project.
9. Having the team reflect at regular intervals on how to become more effective, then tuning and adjusting behavior accordingly.
10. Measuring progress by the amount of completed work.
11. Continually seeking excellence.
12. Harnessing change for a competitive advantage.

**What is join?**

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them.

**Write type of joins.**

Different types of Joins are as follows:

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. FULL JOIN

**A. INNER JOIN**

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.



**B. LEFT JOIN**

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.



**C. RIGHT JOIN**

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.



**D. FULL JOIN**

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.



**Explain working methodology of agile model and also write pros and cons.**

Agile methodology is a project management strategy that divides the project into multiple phases, encouraging continuous improvement for each phase. In the beginning of the project, the team cycles through planning, evaluation and execution stages to collaborate toward multiple project goals. As a methodology, the agile project strategy contains four fundamental values, including:

**Individuals over tools:** Agile methodology values working with individual team members in collaboration efforts rather than relying on tools to accomplish a project. For example, a manager using this strategy may encourage a full team meeting rather than running tests with a software tool in order to solve a process problem.

**Working software over documentation:** While documentation is important in any project for accuracy and accountability purposes, the agile methodology strategy generally prefers real-time reaction to problems rather than continuous documentation. For example, when reacting to a problem in a system, an agile project manager may work to fix the issue before documenting every detail concerning the incident.

**Customer collaboration over contract negotiation:** In agile project management, a group working on a project may prefer to collaborate with other departments or companies rather than negotiate the terms of their contract or promised materials. This is so that they adhere to their set goals more reliably, rather than changing goals to meet projected results.

**Change response over plan rigidity:** While project managers who use agile methodology prefer to not change contract terms, they generally prefer change responses instead of adhering to a plan fully. For example, if the team recognizes that the plan currently in place is not feasible, they change aspects of their process rather than the current goal to help meet needs.

**Pros of agile methodology**

There are many benefits to using an agile methodology strategy in your department, including:

**1. Timely delivery**

Because it focuses on product placement, the agile strategy allows a department to deliver products to clients as quickly as possible. While products and development processes may not be as optimal as possible due to lessened planning stages, the agile method allows improvement to progress through reactionary feedback. For departments that can benefit from producing a stable product first before optimizing the process, agile methodology is an acceptable approach.

**2. Adaptability**

Since the improvement increments between product delivery dates are small, projects can easily change and adapt when underneath an agile strategy system. This allows production to continue at an acceptable rate while improving the process continuously. Departments that work with an agile production system can adapt quickly to change requests, even between clients that have slightly different preferences.

**3. Ease of collaboration**

Because agile working requires much feedback between the client and employees, agile methodology introduces a great system for collaborating between both customers and other departments. In an agile system, management encourages employees to think creatively to solve problems and create solutions for project challenges as they arise. Using an agile system in your department can help your employees learn to collaborate with others to produce a better product with each piece of feedback they receive.

**4. Increased performance improvement**

Because agile department test product increments as they produce them, department members can quickly react to problems as they may arise. The fragmented nature of production and correction allows agile departments to better understand and correct problems quickly. By using feedback from clients and other members, departments can fix issues quickly before the next production cycle.

**5. Transparency**

With the agile approach to departmental work, both potential issues and process improvements become evident with each production cycle. This type of management allows employees to both correct mistakes and improve production details quickly. Because management encourages employees to act independently in this format, the time between understanding an improvement and implementing it may be shorter.

**6. Continuous improvement**

Because the agile department strategy relies on improving a process as it produces products, improvement can directly effect the next product. Additionally, because agile strategies don't stop production to implement fixes, management encourages employees to act on feedback as soon as possible. A continuously improving department can become even more efficient as they sell products.

**7. Higher profits**

The agile department focuses on producing a continuously improving product rather than a perfect one. This enables agile departments to gain profits as quickly as possible, since each product profit brings feedback back to the team. Because an agile department focuses on production, an agile strategy may help a team produce products without stopping production for major fixes or process changes.

**8. Less preparatory work**

Because agile development focuses on the product more than process improvement, an agile department can produce a product more quickly than other management methods. By accepting feedback and making changes over time rather than stopping production to make larger changes, products can improve while not fully compromising improvement efforts. Additionally, when starting a production cycle, the agile departmental values have less preparatory work required before they can begin selling products, saving time between the planning and profit stages.

**Cons of agile methodology**

There are some disadvantages from using the agile methodology style of project management, including:

**1. Transfer difficulties**

When switching from one management style to another for an entire department, it may take time to fully acclimate to the new responsibilities and style of how projects progress within the system. Leaders may have some initial difficulties adjusting to the management style, although consistent practice and training can help all employees adjust to new tactics. While agile management encourages employees to act independently toward goals, consider dividing departments into groups for the first few objectives of the project. This is so that each member can consult another concerning questions or transfer difficulties.

**2. Variable goals**

Because the agile methodology focuses on multiple goals at a time, certain goals may not receive as much focus throughout the entire project. Variable goals within a department can lead to a lack of specific goals for a team, which may lead to unknown deadlines and less accurate costs. One way to prevent variable goals from allowing unwarranted costs or missed deadlines can be to schedule regular meetings to discuss goals. Another way to unify goal costs is to set a policy regarding spending in the department so that cash flow follows a policy to reduce unforeseen spending.

**3. Lack of documentation**

In the agile methodology system, documentation is less important than reactionary planning and progress. This can make documentation responsibilities such as record keeping, billing statements and plan-keeping complete more slowly in comparison to other duties. To help improve documentation while using this methodology, consider taking record of documented statements at least once a month. If documentation has fallen behind, emphasizing it on the schedule can help bring it back to proper importance again.

**4. Less documented improvement**

Because agile methodology strategies focus on reactionary improvement rather than documented improvement phases, documented success and calculated methodology strategies may be ineffective throughout the project. While reactionary tactics can be effective for improving a process or product, consider improving your team's documented improvement efforts by scheduling document checks regularly. Keeping a gradual record of success and progress can help the team gain hindsight concerning objectives and next steps.

**5. Goal focus shifting**

Since agile methodology involves focus shifting based on which part of a project requires the most attention, it may be difficult to lead all team members toward a singular goal. In later stages of the project, it may become preferable to have the majority of department members focusing on a singular goal. To help achieve goal unity, consider allocating goal-related tasks to multiple members of the department during the final phases of the project. Unity through allocation can help a department work as a unified force, even if only temporarily.

**6. Less predictability**

Because agile production relies on continuous improvement and customer feedback, the department may not be able to predict profits before production begins. Agile production focuses on sending acceptable products to customers as soon as possible, which can make potential issues harder to predict over time. However, using feedback effectively and quickly during the process may be able to increase the product's fault predictability marginally over time, especially if the department documents issues during the process.

**Draw usecase on Online shopping product using COD.**

**Add category**

**Registered Customer**

**Add item**

**Add to cart**

**Proceed to buy**

**Web Customer**

**View items**

**Add address**

<<include>>

**Select payment method**

**Select COD**

**New customer**

**Place your order**

<<include>>

**Draw usecase on Online shopping product using payment gateway.**

