# EXPERIMENT-1

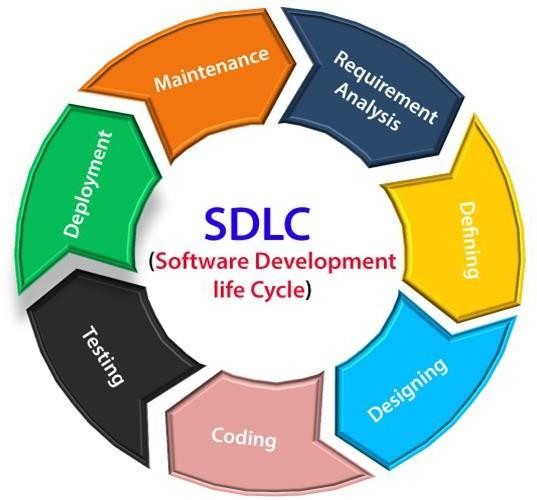
**AIM: To get an understanding of the stages in software development lifecycle, the process models, values and principles of agility and the need for agile software development. This will enable you to work in projects following an agile approach to software development.**

Software Development Life Cycle:

A software life cycle model (also termed process model) is a pictorial and diagrammatic presentation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be under taken.

In other words, a lifecycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

SDLC Cycle represents the process of developing software. SDLC framework includes the following steps:



# The stages of SDLC are as follows:

## Stage1: Planning and requirement analysis

Requirement Analysis is the most important and necessary stage in SDLC.

The senior members of the team perform it with inputs from all the stakeholders and do main experts or SMEs in the industry.

Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.

For Example, A client wants to have an application which concerns money transactions. In this method, the requirement has to be precise like what kind of operations will be done, how it will be done, in which currency it will be done, etc.

Once the required function is done, an analysis is complete with auditing the feasibility of the growth of a product. In case of any ambiguity, a signal is setup for further discussion.

Once the requirement is understood, the SRS (Software Requirement Specification) document is created. The developers should thoroughly follow this document and also should be reviewed by the customer for future reference.

## Stage2: Defining Requirements

Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stake holders.

This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project lifecycle.

## Stage3: Designing the Software

The next phase is about to bring down all the knowledge of requirements, analysis and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.

## Stage4: Developing the project

In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters,

debuggers, etc. are used to develop and implement the code.

## Stage5:Testing

After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage. During this stage, unit testing, integration testing, system testing, acceptance testing are done.

## Stage6: Deployment

Once the software is certified, and no bugs or errors are stated, then it is deployed. Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment. After the software is deployed, then its maintenance begins.

## Stage7: Maintenance

Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time. This procedure where the care is taken for the developed product is known as maintenance.

**Disadvantages Of System Development Life Cycle**

**Several approaches are not flexible**. It may be hard to estimate the overall costs at the start of the software project. Performing testing at the end of the development process can slow down development teams.

## What is a software process model?

A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed.

# A model will define the following:

* The tasks to be performed
* The input and output of each task
* The pre and post conditions for each task
* The flow and sequence of each task

There are many kinds of process models for meeting different requirements. We refer to these as **SDLC models** (Software Development Life Cycle models). The most popular and important SDLC models are as follows:

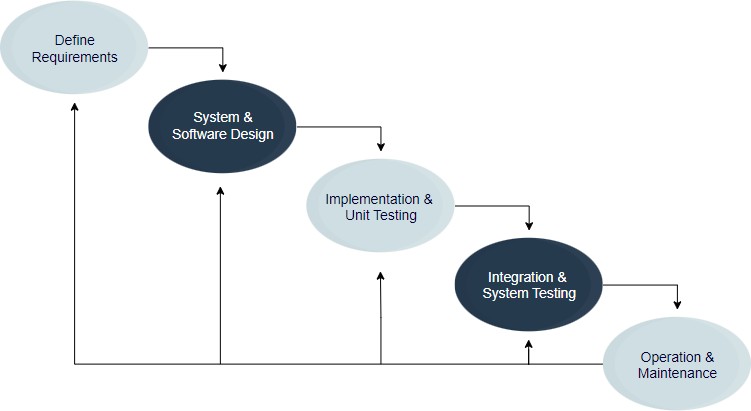
Waterfall model

* V model
* Incremental model
* RAD model
* Agile model
* Iterative model
* Prototype model
* Spiral model

**Waterfall Model**

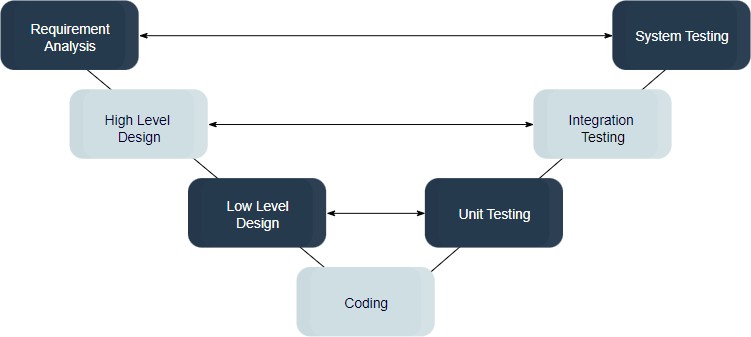
The waterfall model is a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project. Each activity in the waterfall model is represented as a separate phase arranged in linear order.

It has the following phases:

* Requirements
* Design
* Implementation
* Testing
* Deployment
* Maintenance

# V Model

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

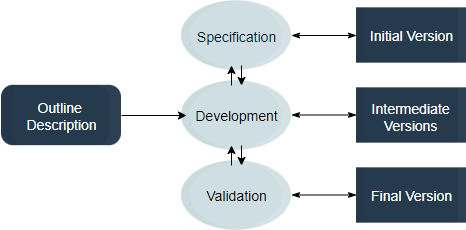


# Incremental Model

The incremental model divides the system’s functionality into **small increments** that are delivered one after the other in quick succession. The most important functionality is implemented in the initial increments.

The subsequent increments expand on the previous ones until everything has been updated and implemented.

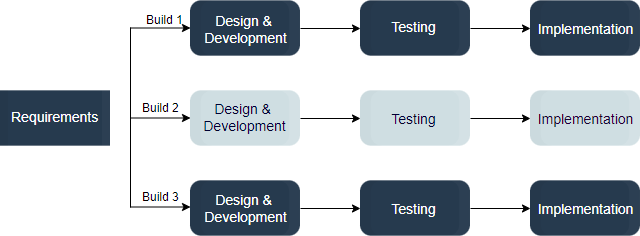
Incremental development is based on developing an initial implementation, exposing it to user feedback, and evolving it through new versions. The process’ activities are interwoven by feedback.



# Iterative Model

The iterative development model develops a system through **building small portions** of all the features. This helps to meet initial scope quickly and release it for feedback.

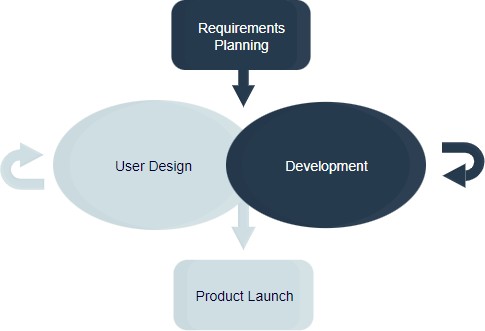
In the iterative model, you start off by implementing a small set of the software requirements. These are then **enhanced iteratively** in the evolving versions until the system is completed. This process model starts with part of the software, which is then implemented and reviewed to identify further requirements.



# RAD Model

The Rapid Application Development (RAD model) is based on iterative development and prototyping with **little planning involved**. You develop functional modules in parallel for faster product delivery. It involves the following phases:

1. Business modeling
2. Data modeling
3. Process modeling
4. Application generation
5. Testing and turnover

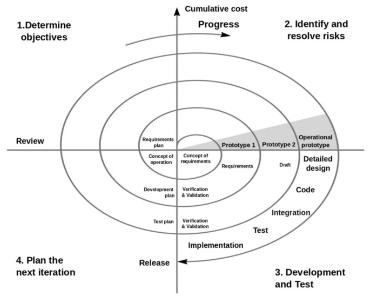


# Spiral Model

The spiral model is a risk driven iterative software process model. The spiral model delivers projects in loops. Unlike other process models, its steps aren’t activities but **phases** for addressing whatever problem has the greatest risk of causing a failure.

You have the following phases for each cycle:

1. Address the highest-risk problem and determine the objective and alternate solutions
2. Evaluate the alternatives and identify the risks involved and possible solutions
3. Develop a solution and verify if it’s acceptable
4. Plan for the next cycle



# Agile model

The agile process model encourages **continuous iterations of development** and testing. Each incremental part is developed over an iteration, and each iteration is designed to be small and manageable so it can be completed within a few weeks.

Each iteration focuses on implementing a small set of features completely. It involves customers in the development process and minimizes documentation by using informal communication.

Agile development considers the following:

* Requirements are assumed to change
* The system evolves over a series of short iterations
* Customers are involved during each iteration
* Documentation is done only when needed

Though agile provides a very realistic approach to software development, it isn’t great for complex projects. It can also present challenges during transfers as there is very little documentation. Agile is great for projects with **changing requirements**.

Some commonly used agile methodologies include:

* **Scrum:** One of the most popular agile models, Scrum consists of iterations called sprints. Each sprint is between 2 to 4 weeks long and is preceded by planning. You cannot make changes after the sprint activities have been defined.
* **Extreme Programming (XP):** With Extreme Programming, an iteration can last between 1 to 2 weeks. XP uses pair programming, continuous integration, test-driven development and test automation, small releases, and simple software design.
* **Kanban:** Kanban focuses on visualizations, and if any iterations are used they are kept very short. You use the Kanban Board that has a clear representation of all project activities and their numbers, responsible people, and progress.

**What are agile methodologies?**

The first we heard about Agile development (or the term Agile) was back in 2001 when a community of developers that had grown tired of using development methods considered to be

―heavy‖ – namely, the waterfall model – decided to set out a manifesto: The Agile Manifesto. This

very important document had so much impact that It has worked as a bible for Agile development even until today, laying out all principles and good practices.

Agile methodologies argue that, above all, **we should seek client satisfaction through the continuous delivery of value-adding software, by staying in constant communication with the client, and also by focusing on communication between team members.** Contrary to previous practices, Agile methodology is not characterized by the complete definition of a product, but rather

―step by step‖– a complete analysis or the definition of all categories/ requirements, by dynamic interaction that allows constant delivery

– Focusedon―near-shore‖visibilitybutneverlosingthelong-termproductgoal.

Customer satisfaction is the top priority, demonstrated through continuous delivery and added value.

Changes to requirements should be accepted– rather than pursuing ―rigid‖ requirements, even if at a late stage in development: ―Agile processes harness change for the customer’s competitive advantage‖, asisstated in the Manifesto.

The customer and the development team should work together on a daily basis, facilitating team and product synchronization.

It is vital to provide a pleasant environment and good support to development teams. Only in this way is it possible to keep them motivated.

Agile processes promote sustainable development due to their constant rhythm and technical excellence, which in turn improves productivity.

Retrospective moments inside a team are essential, allowing it to make the necessary adjustments and promote efficiency.

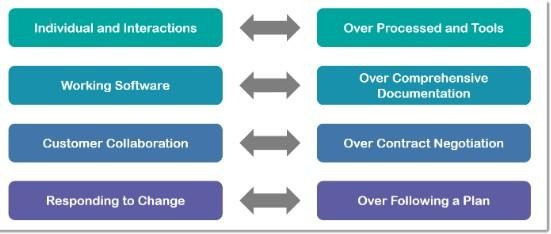
Basically, Agile development follows an incremental model, which develops collaboration with in the team and continuous planning, as well as everlasting evolution and learning. Agile methodologies should respect the software development cycle – planning, execution and final delivery – therefore allowing software to be developed in stages; this makes it easier to identify and resolve bugs or new needs.

The main advantage of using Agile methodologies is not just the fast delivery of software, but also the constant delivery of ―value‖ to the customer, since deliveries are incremental.

There are countless methodologies that follow this Agile mindset. In this blog post, we highlight the **five main Agile methodologies** and their advantages and disadvantages in the software development universe. But we cannot dive into the different methodologies with out first referring to the growing popularity of the Agile methodologies (or some of them at least) with business management. This proves that is just not software development that can be enriched using these practices.

Business development is becoming an increasingly unpredictable playground, just like software development. So the challenge is now why not implement Agile methodologies in business process management? **Agile methods are adaptable**, allowing for rapid decision-making and instant influence on business development.

**4 values of Agile Manifesto**



1. **Individuals and interactions over processes and tools**

This value of the Agile manifesto **focuses on giving importance to communication with the clients**.There are several things aclient may want to ask and it is the responsibility of the team members to ensure that all questions and suggestions of the clients are promptly dealt with.

# Working product over comprehensive documentation

In the past, more focus used to be on proper documentation of every aspect of the project. There were several times when this was done at the expense of the final product. The Agile values dictate that **the first and fore most duty of the project team is completing the final deliverables** as identified by the customers.

# Customer collaboration over contract negotiation

Agile principles **require customers to be involved in all phases of the project** .The [Water fall](https://kissflow.com/project/agile/traditional-vs-agile-project-management/) [approach](https://kissflow.com/project/agile/traditional-vs-agile-project-management/) or Traditional methodologies only allow customers to negotiate before and after the project. This used to result in wastage of both time and resources. If the customers are kept in the loop during the development process, team members can ensure that the final product meets all the requirements of the client.

# Responding to change over following a plan

Contrary to the management methodologies of the past, Agile values are against using elaborate plans before the start of the project and continue sticking to them no matter what Circumstances change and some times customers demand extra features in the final product that may change the [project scope.](https://kissflow.com/project/project-scope-management/) In these cases, project managers and their **teams must adapt quickly in order to deliver a quality product and ensure 100% customer satisfaction**.

**12 Principles of Agile Manifesto.**

1. **Customer satisfaction through continuous delivery of the product**

In the case of [traditional management methodologies,](https://kissflow.com/project/agile/traditional-vs-agile-project-management/) customers get to see the product only after completion and when several tests and quality checks have been performed. This not only keeps the customers in dark but also make sit problematic for the team members to introduce any changes in the product.

In order to keep the customers happy, it’s important to continuously engage them with a working version of the product. Show small increments every [sprint planning](https://kissflow.com/project/agile/scrum-sprint-planning/) and make changes as required.

# Divide large chunks of work into smaller and achievable tasks for quicker completion and easier integration of changes

Handling a huge and complex task would be both time and energy-consuming while [managing project tasks](https://kissflow.com/project/team/guide-to-task-management/). A better way is to divide the task into smaller parts that can be easily

completed. The customers would always be kept in the loop and it would be easier for the team members to identify potential bottlenecks and handle any potential delays.

# Adhere to the decided time frame for the delivery of a working product

The Agile philosophy favors a smaller time frame and delivers working software frequently. This iterative process requires team members to continuously improve their performance.

# All stake holders must frequently collaborate to ensure that the project is going in the correct direction

A major problem associated with traditional project management methodologies is that the [projects take holders](https://kissflow.com/project/project-stakeholder-management/) are often oblivious to the development stages of the project. The Agile principles encourage all stake holders to remain involved in all stages of the project in order to ensure constant feedback and a valuable end product.

# Create a supportive environment to motivate team members and encouraging them to get the job done

It is the responsibility of the project manager to create motivating environment and support where members are not afraid to voice their opinions and give suggestions for the betterment of the team’s performance. Giving the team members the confidence and motivating them would cause their overall performance to improve significantly which will ultimately be better for the project.

# Preferface –to -face communication over other methods

In the Agile manifesto, a lot of importance is given [to effectiv ecommunication](https://kissflow.com/project/communication-in-project-management/) between the involved parties. For effective communication, methods like memos and email are not preferred and more importance is given to face –to –face communication. This is now easier because of the advances in communication technologies. All stake holders can now have a meeting through video conferencing rather than coming to the office for a short meeting.

# Working software is the primary measure of progress

The only factor to measure success is the delivery of a working product that satisfies the customer. Before Agile, there were many measures of success and that resulted in a drop in the quality of [the](https://kissflow.com/project/project-deliverables/) [final product.](https://kissflow.com/project/project-deliverables/)

# Try to maintain a constant pace of development

A repeatable and iterative pattern should be established where sustainable development of the project takes place at a constant rate. This prevents the team members from being over whelmed and

over-stressed, while the project keeps on progressing at an acceptable rate.

# Maintain the quality of the product by paying attention to technical details

Providing value to the customer is the primary objective of any [Agile team](https://kissflow.com/project/agile/agile-team/). It’s extremely important to have a multi-skilled team that can handle all the technical aspects of the project and provides the opportunity for continuous improvement.

# Maintainsimplicity

In each time box, the tasks at hand should be the main focus of all team members. Too much planning and adding extra features to the product should be avoided during the development.

# Promote self – organization in the team

A [self – organized team](https://kissflow.com/project/agile/importance-of-self-organizing-teams/) with decision –m aking powers would simply perform better because the responsibility of satisfying the customers will on the team members, rather than a single project manager.

# Regularly reflect on your performance for continuous improvement

Agile methodologies stand on the concept to fiteration, where teams learn from their past mistakes and continuously improve their performance. Project managers should promote sessions where the whole team reflects on their performance and discuss ways to improve etheir technical and management skills.

**Need for agile software development**