Devops-Unit-1 - MATERIAL

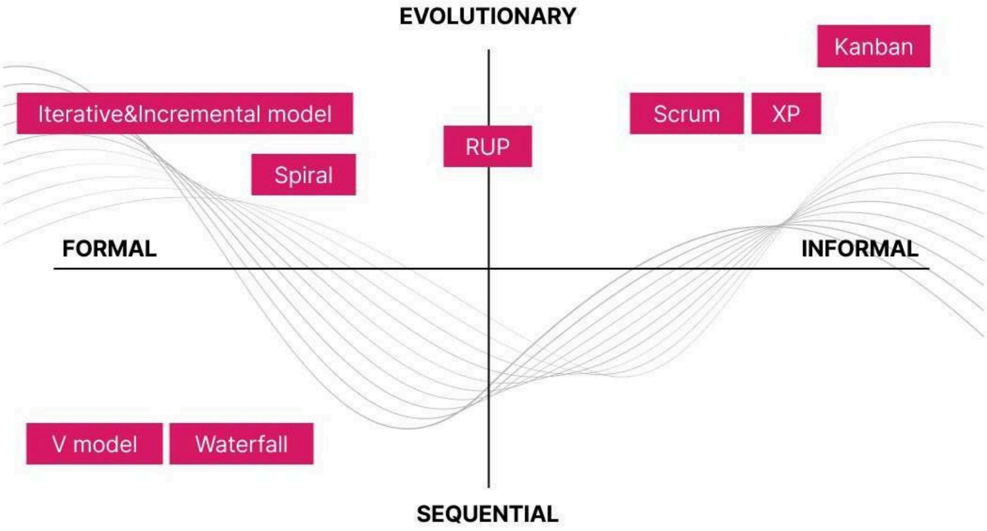
COMPUTER SCIENCE ENGINEERING (Jawaharlal Nehru Technological University, Kakinada)

UNIT- I

Phases of Software Development life cycle. Values and principles of agile software development.

What Is an SDLC?

A software development life cycle (SDLC) is a methodology followed to create high-quality software. By adhering to a standard set of tools, processes, and duties, a software development team can build, design, and develop products that meet or exceed their clients' expectations.



Phases of the Software Development Life Cycle

SDLC processes generally number at 6 distinct stages: planning, analysis, designing, development and testing, implementation, and maintenance. Each of them is briefly explained below.

1. Planning

The very first phase of the SDLC starts with requirement gathering. This is known as the planning stage. It is the most important phase of the entire SDLC from the perspective of project managers and stakeholders.

The planning stage answers such questions as:

- How will the software be used?

- What data will serve as the input of the software?
- What will be the output of the software?
- Who is going to use the software?

2. Analysis

Once all the requirements are known, it's time to analyze them for feasibility and validity. The team decides whether it will be possible to add the requirements to the software. After that, a Requirement Specification document is designed. This serves as the guide for carrying out the next phase of the SDLC.

3. Designing

This stage includes the designing of requirements specified in the very first phase of the SDLC. In addition to assisting in specifying hardware and system requirements, that stage also helps define the overall software architecture.

The system design specifications prepared in the designing phase serve as the input for the next i.e. fourth stage of the SDLC. During the designing phase, testers are required to form an apt testing strategy. It contains what needs to be tested, and how it needs to be tested.

4. Development & Testing

Some development teams consider development and testing to fall under a single-phase, others prefer to break it into two sub-phases. Irrespective of the choice a development team makes, the whole process remains the same. It's all a matter of preference.

Once the system design documentation is complete, the whole task is divided into modules or units. After that, the actual coding begins.

Because this phase includes coding, it is the most important phase of the SDLC for the developer team. Moreover, this is the longest phase of the entire software development lifecycle. Once the code is fully developed, testing is carried out against the requirements.

Testing includes functional testing, such as acceptance testing, integration testing, system testing, and unit testing, as well as non-functional testing.

5. Implementation

Also known as the deployment phase, the implementation phase occurs right after the successful testing of the software product. It is focused on delivering the software to the end-user or installing it onto the customer's system(s).

The first thing that takes place once the product is delivered to the customer is beta testing. All bugs and enhancements are then reported to the developer team working on the project. Once the changes are complete, the final deployment takes place.

6. Maintenance

Finally, there's the maintenance phase, which occurs over time after the product has been released. This phase deals with dealing with problems experienced by the customers/end-users while using the software.

Advantages and Disadvantages of SDLCs

Advantages	**Disadvantages**
Ample documentation	Unsuitable when there are many iterations and/or regular reviews
Comprehensive steps	Difficult to estimate costs and project overruns
Easy maintenance	Inflexible when it comes to changes in requirements
Effective development and design standards	Increased development cost and development time
Sets reliable evaluation costs and completion targets	Limited automation of documents and standards
Can monitor full-scale projects	Little parallelism
Strong control	The software is required to be thoroughly defined before the start
Precise user input	User input might be limited in certain scenarios
Tolerates changes to MIS in staffing	

What is the best SDLC methodology?

There is no straight answer to this question, as the software life cycle model you choose will depend on the size and scope of the software you are building, on top of other factors. The best SDLC methodology will always come down to the requirements and project context.

There are also alternatives to the types of SDLC methodologies detailed further below. The most popular one is the RAD or Rapid Application Development. RAD offers implementation of CASE tools, joint application development, and prototyping. The advantages of RAD are active user involvement, a faster approach, and reduced development cost.

The main SDLC models include:

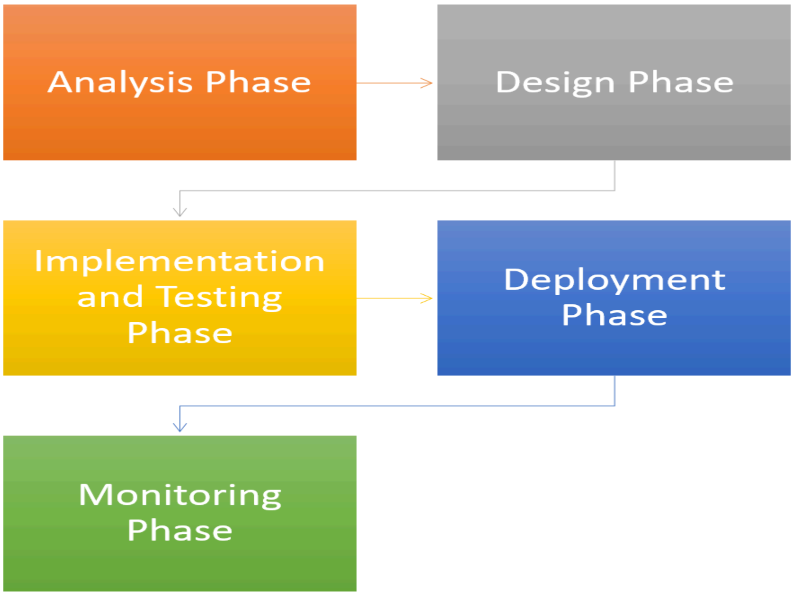
- Waterfall: Follows a sequential model of phases, each of which has its own tasks and objectives

- Cleanroom: A process model that removes defects before they cause serious issues
- Incremental: Requirements are divided into multiple standalone modules
- V-Model: Processes are executed sequentially in a V-shape (each step comes with its own testing phase)
- Prototyping: A working replication of the product is used to evaluate developer proposals
- Big Bang: Requires very little planning and has no formal procedures; however, it's a high- risk model
- Agile: Uses cyclical, iterative progression to produce working software

The last one on the list, **Agile**, is what we're focusing on today. You see, a traditional SDLC (like Waterfall) front-loads the work – so if you require a large product, it can take a long time before the team even builds a working prototype. Most software development startups don't have the financial means to wait that long. Well-funded competitors could beat them to the market; so, to make processes more rapid without compromising on quality, many development companies are embracing the Agile software development lifecycle.

1. Waterfall model:

The waterfall model uses a sequential approach to building any software. Below is the phases waterfall model as follows:



- The model has various phases starting with the Analysis phase. In this phase, you try to analyze the requirements given by the clients
- In the Design phase, the analysis will be transformed into an ordered structure, i.e., you prepare a blueprint of the software
- After completing the design phase, it's time for developers to get their hands dirty by coding for the application. This would be the source code, and you test it in the testing process. Different checks on the application are performed, such as unit testing, integration testing, performance testing, etc
- In the Deployment phase, the tested application is deployed onto production servers
- Finally comes the Monitoring phase, wherein 60 percent of the entire effort is spent monitoring the application's performance

Let's see its advantages and disadvantages

Advantages:

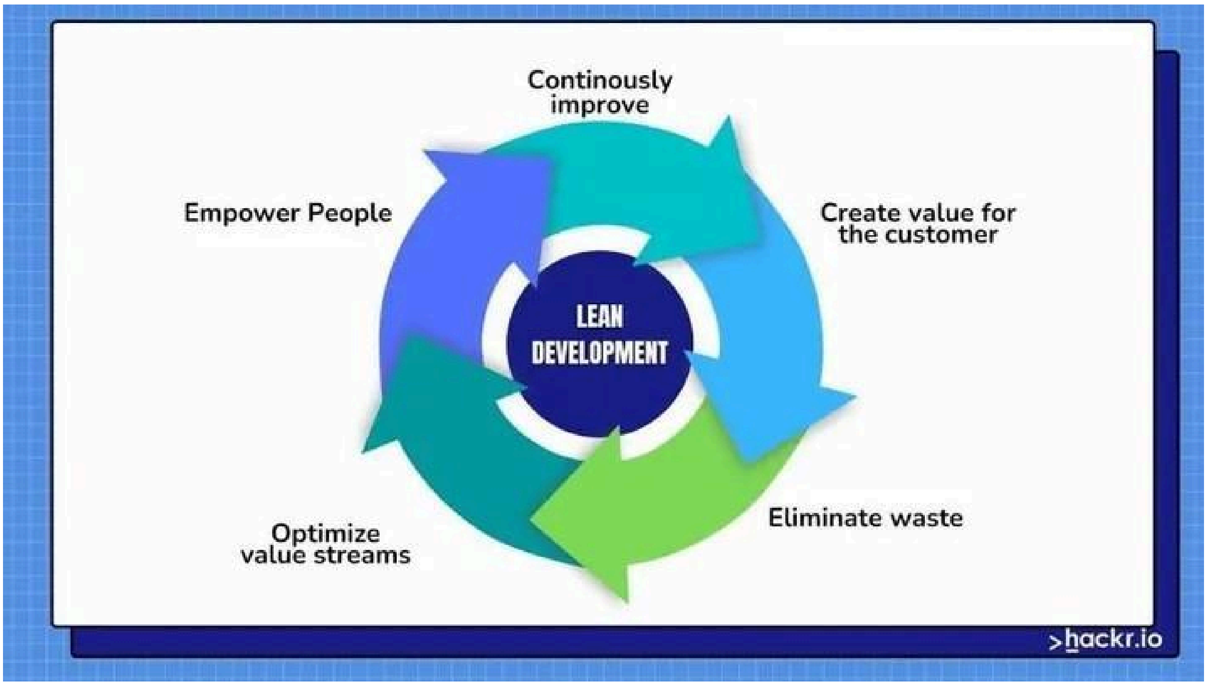
- Easy to use and simple to understand
- As it's a top-down approach process, overlapping of the phases is avoided
- Suitable for small-projects
- Easy testing and analysis

Disadvantages:

- Changes to the product are complicated to draw while it is in the testing stage
- Not suitable for complex projects
- The end product is only available at the end of the cycle

To summarize, the Waterfall model was only acceptable for projects with stable specifications. By stable, I mean that over time, requirements will not change. But this is an improbable thing in today's world, as specifications keep changing from time to time.

2. Lean Software Development Life Cycle



The Lean methodology takes inspiration from lean manufacturing practices and principles. It follows a set of seven principles, which are:

- Eliminate waste
- Amplify/refine learning
- Decide as late as possible
- Deliver as fast as possible
- Empower the team
- Emphasize conceptual integrity
- See the whole/Operating from the top-level

Project teams working on the Lean model aim at finding opportunities to cut waste at every step of the SDLC process. Typically, this includes skipping unimportant meetings and reducing documentation.

In actuality, the Lean methodology is very much similar to the Agile methodology with some noteworthy differences.

The most important distinction between the two SDLC methodologies lies in terms of prioritizing customer satisfaction. The Agile model makes customer satisfaction a priority from the very beginning. Consequently, the project teams involved respond instantly to stakeholder feedback throughout the SDLC procedure.

On the other hand, Lean methodology gives the topmost priority to the elimination of waste. This is done in order to create more overall value for the clients.

Pros:

- Applicable across team boundaries, doing well to integrate teams and optimize collaboration. Thus, it works well in line with the Agile and DevOps methodologies

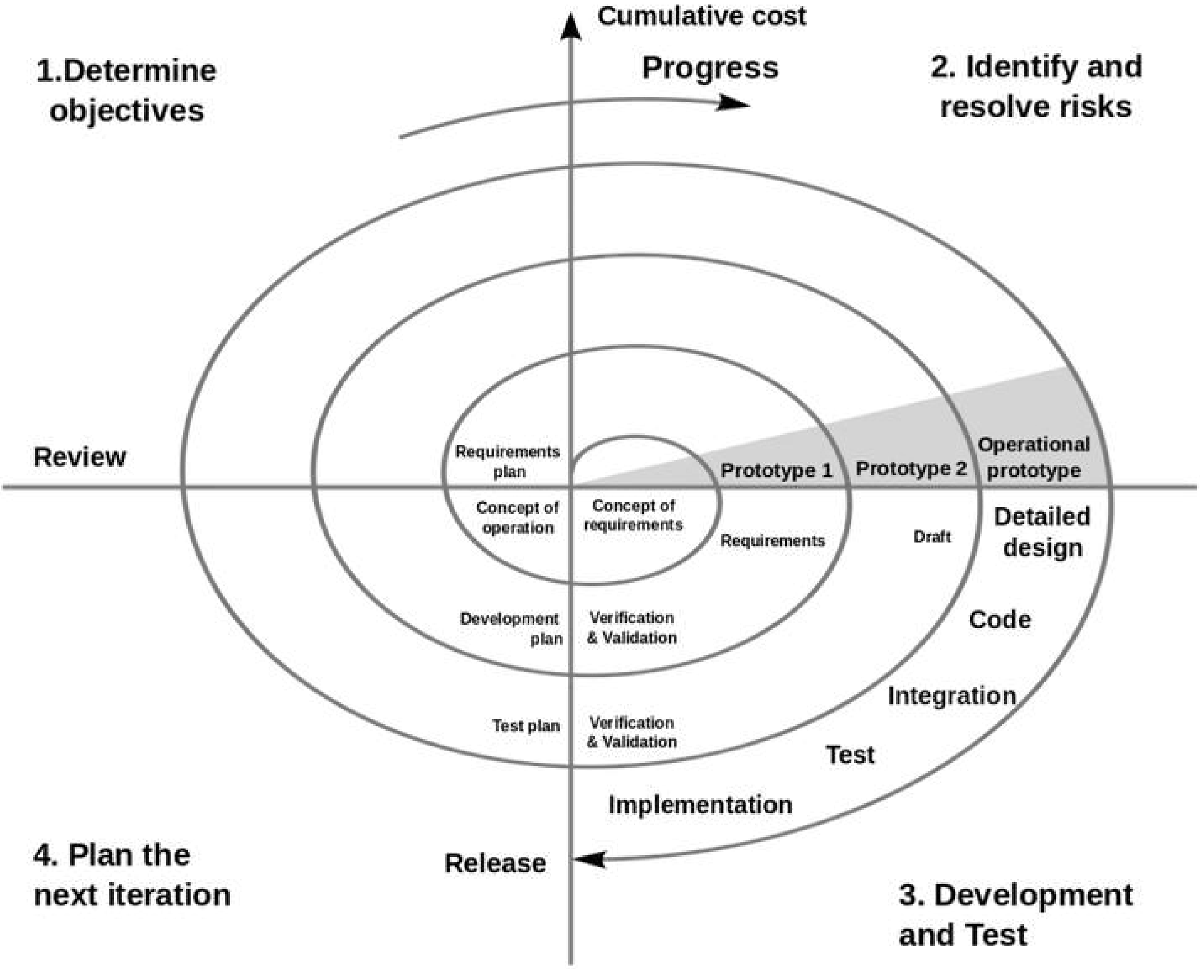
- Facilitates delivering greater functionality in a shorter span of time

- Easily scalable, making it an apt fit as an alternative to contemporary SDLC methodologies designed for large, complex projects
- Empowers the project development team in terms of the decision-making process. As a result, it improves motivation
- Saves cost and time by eliminating unessential activities

Cons:

- Demands excellent documentation, especially with respect to business requirements. Failing to do it might result in underdeveloped or wrongly developed areas pertaining to insufficient documentation
- Heavily team dependent. This means it is essential to put together an experienced team with a high skill level
- Relatively easy to lose focus

Spiral Software Development Life Cycle



The Spiral methodology is considered one of the most flexible SDLC models. Typically adopted for full-blown projects, the Spiral model lets development teams build a highly customized product.

Spiral methodology passes through four phases repeatedly until the project is completed. This allows for following multiple rounds of product refinement. These four phases are planning, risk analysis, engineering, and evaluation.

Each iteration of the Spiral methodology begins with predicting potential risks and the best way to avoid or mitigate them.

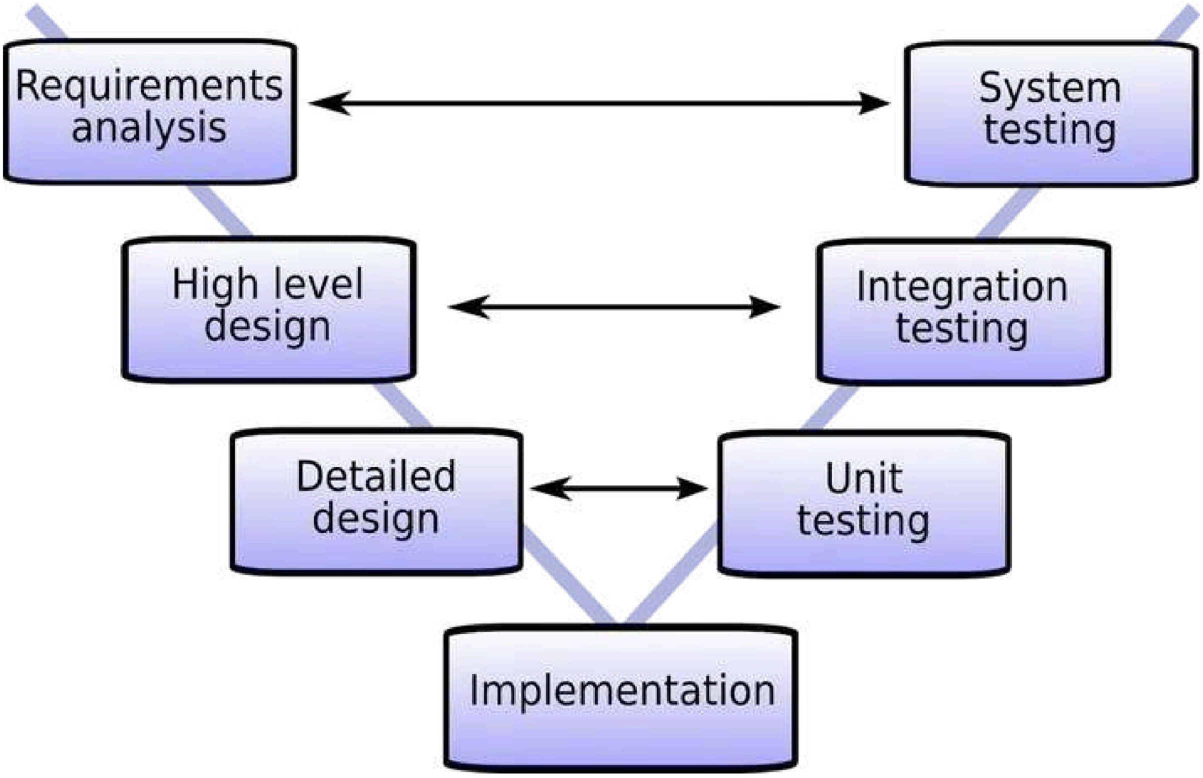
Pros:

- Can accommodate new changes or functionality at later stages of development
- Cost estimation becomes as the prototype build is done in small increments
- Better risk management with repeated development
- Emphasizes customer feedback
- Faster development and systematic addition of features

Cons:

- Demands risk management expertise
- High risk of not meeting budget or schedule deadlines
- Impractical for small projects
- Much more documentation due to intermediate phases

3. V Software Development Life Cycle



V Model stands for Verification and Validation Model. Though inspired by the Waterfall model, where the testing is done at the end of the project, it differs in that it introduces testing at every stage of development.

Similar to the Waterfall model, each next stage of the V model begins only when the previous one has been completed.

Pros:

- Ability to avoid the downward flow of defects
- An apt fit for small projects with easily understandable requirements
- Higher chances of success
- Offers ease and simplicity
- Proactive tracking of potential defects
- Saves a lot of time as planning and designing related to testing is done before the actual coding takes place

Cons:

- Even more rigid than the Waterfall model
- No early prototype creation is possible as the software is developed during the implementation phase
- Requirement and test documents need to be updated if changes are made during the development

The Four Values of The Agile Manifesto

The Agile Manifesto is comprised of four foundational values and 12 supporting principles which lead the Agile approach to software development. Each Agile methodology applies the four values in different ways, but all of them rely on them to guide the development and delivery of high-quality, working software.

1. **Individuals and Interactions Over Processes and Tools**

   The first value in the Agile Manifesto is “Individuals and interactions over processes and tools.” Valuing people more highly than processes or tools is easy to understand because it is the people who respond to business needs and drive the development process. If the process or the tools drive development, the team is less responsive to change and less likely to meet customer needs. Communication is an example of the difference between valuing individuals versus process. In the case of individuals, communication is fluid and happens when a need arises. In the case of process, communication is scheduled and requires specific content.

2. **Working Software Over Comprehensive Documentation**

   Historically, enormous amounts of time were spent on documenting the product for development and ultimate delivery. Technical specifications, technical requirements, technical prospectus, interface design documents, test plans, documentation plans, and approvals required for each. The list was extensive and was a cause for the long delays in development. Agile does not eliminate documentation, but it streamlines it in a form that gives the developer what is needed to do the work without getting bogged down in minutiae. Agile documents requirements as user stories, which are sufficient for a software developer to begin the task of building a new function.The Agile Manifesto values documentation, but it values working software more.

3. **Customer Collaboration Over Contract Negotiation**

Negotiation is the period when the customer and the product manager work out the details of a delivery, with points along the way where the details may be renegotiated. Collaboration is a different creature entirely. With development models such as Waterfall, customers negotiate the requirements for the product, often in great detail, prior to any work starting. This meant the customer was involved in the process of development before development began and after it was completed, but not during the process. The Agile Manifesto describes a customer who is engaged and collaborates throughout the development process, making. This makes it far easier for development to meet their needs of the customer. Agile methods may include the customer at intervals for periodic demos, but a project could just as easily have an end-user as a daily part of the team and attending all meetings, ensuring the product meets the business needs of the customer.

4. **Responding to Change Over Following a Plan**

Traditional software development regarded change as an expense, so it was to be avoided. The intention was to develop detailed, elaborate plans, with a defined set of features and with everything, generally, having as high a priority as everything else, and with a large number of many dependencies on delivering in a certain order so that the team can work on the next piece of the puzzle.

With Agile, the shortness of an iteration means priorities can be shifted from iteration to iteration and new features can be added into the next iteration. Agile's view is that changes always improve a project; changes provide additional value.

Perhaps nothing illustrates Agile's positive approach to change better than the concept of Method Tailoring, defined in An Agile Information Systems Development Method in use as: "A process or capability in which human agents determine a system development approach for a specific project situation through responsive changes in, and dynamic interplays between contexts, intentions, and method fragments." Agile methodologies allow the Agile team to modify the process and make it fit the team rather than the other way around.

The Twelve Agile Manifesto Principles

The Twelve Principles are the guiding principles for the methodologies that are included under the title "The Agile Movement." They describe a culture in which change is welcome, and the customer is the focus of the work. They also demonstrate the movement's intent as described by Alistair Cockburn, one of the signatories to the Agile Manifesto, which is to bring development into alignment with business needs.

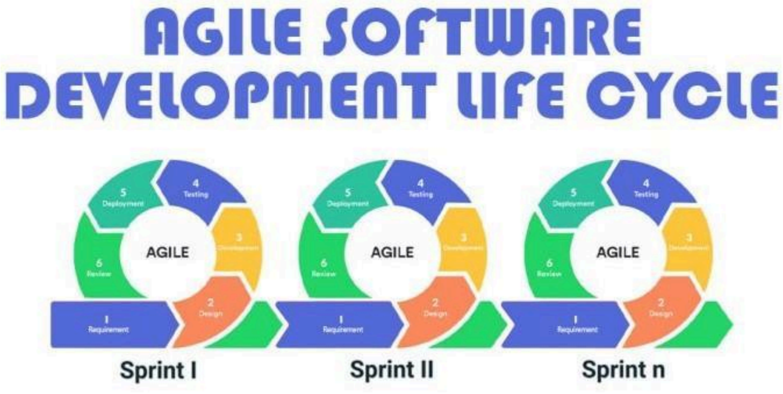
The twelve principles of agile development include:

1. **Customer satisfaction through early and continuous software delivery** – Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.
2. **Accommodate changing requirements throughout the development process** – The ability to avoid delays when a requirement or feature request changes.

3. **Frequent delivery of working software** – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.
4. **Collaboration between the business stakeholders and developers throughout the project** – Better decisions are made when the business and technical team are aligned.
5. **Support, trust, and motivate the people involved** – Motivated teams are more likely to deliver their best work than unhappy teams.
6. **Enable face-to-face interactions** – Communication is more successful when development teams are co-located.
7. **Working software is the primary measure of progress** – Delivering functional software to the customer is the ultimate factor that measures progress.
8. **Agile processes to support a consistent development pace** – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.
9. **Attention to technical detail and design enhances agility** – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
10. **Simplicity** – Develop just enough to get the job done for right now.
11. **Self-organizing teams encourage great architectures, requirements, and designs** – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.
12. **Regular reflections on how to become more effective** – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

The intention of Agile is to align development with business needs, and the success of Agile is apparent. Agile projects are customer focused and encourage customer guidance and participation. As a result, Agile has grown to be an overarching view of software development throughout the software industry and an industry all by itself

The Phases Of Agile Software Development Life Cycle&Workflow And Project Management



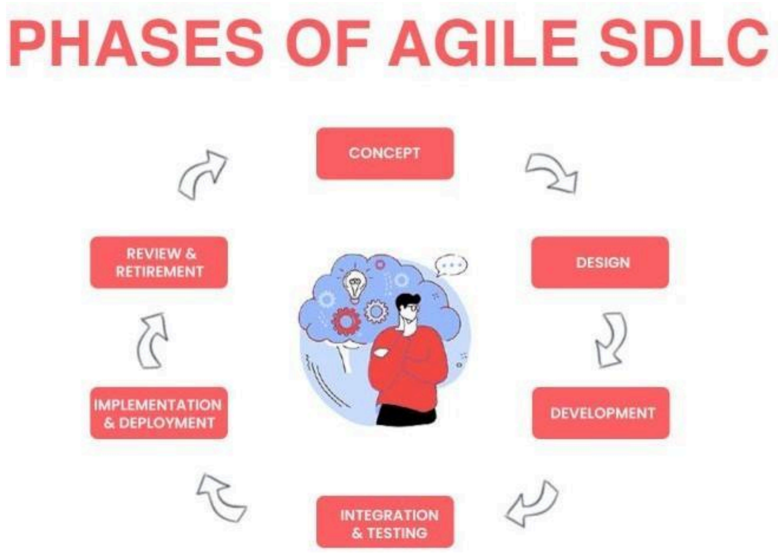
What is Agile model in the **software development life cycle**? Agile software development life cycle is a method or process that focuses on building and maintaining software. It's a scope detailing every stage from beginning to completion and further modification/ upgrading & maintenance of software.

SDLC is the short form of this significant useful term which applied for building high-quality software on budget & on time.

Moreover, the software development life cycle in Agile is a bit similar term of **project management functionally**. Therefore, if you want to get a clear concept of Agile SDLC models, you first need to know about their phases.

Now it's the time to reveal the phases, Agile SDLC workflow, benefits of Agile methods & the effective **Agile process** for your software development.

Phases of Agile Software Development Life Cycle



Phase 1: Concept & Requirement

Requirement gathering is the first phase of the Agile development cycle. In this stage, stakeholders determine their basis of requirement & overall project assessment. After that, the developing team draws the solution by requirement analysis.

The following questions are a reflection of the performance of this stage as the basic activities of the **agile team**.

- Who will use the software?
- How will they use the software?
- What would the required software need as input?
- What would the required software give out as final output?

Gathering every requirement and analysis on that the product owner calculates the risk of software development, set some achievable goals, time frame, product functionality etc.

As the outcome of the requirement phase, documentation development of the functional requirement & budget release is also performed in this stage of development.

Phase 2: Design

The stakeholder introduces the requirement outline to the **software development** team in the design phase.

After entirely knowing the expectation of the stakeholders, the developer team make plans and basic frameworks for the project. The team discusses the product functionality, programming language, essential tools, etc.

In the design stage, the team remains engaged in things like:

- The layout of the mobile, desktop, and various devices for the website
- Color scheme fix according to the project
- What programming languages should use
- Backend frameworks designs
- System server designs

This phase aims to create the agile framework as the main base of the software architecture to work on.

Phase 3: Development

Next is the development phase, also known as the iteration phase of **Agile lifecycle**. The development phase takes the maximum amount of work and time on the Agile development lifecycle. But this phase is the part where the actual fun begins!

In this phase, the developer team works with the UX designer and user stories to combine all the requirements and turn these into code.

Let's get the story of this phase here:

- The development team initiates the building code.
- Sets up the hardware.
- Configures the servers.
- The designers are working on improving the user interface & user experience.
- The testers analyze the requirements.
- Start designing test plans.

After all, the software development team gets the flashlight here where they do the max of the work of an Agile project!

Phase 4: Integration & Testing

Testing is the most vital phase of the Agile application development lifecycle. In this phase, the software gets ready to release. But before release, the development team conducts a couple of tests to ensure the software functionalities.

So that they test the chunk of code is clean or not, software process adaptability, product backlog, features competitive advantages etc. If there is any error or bugs, the developer team will fix it with immediate action & ensure secure software.

Get the following points to better understand for this testing phase:

- Brainstorm on every possible test case
- Use the test cases according to the built feature
- Incorporate them to create a 360-degree test plan that can detect all of the bad features & existing bugs.
- Compiled every planned test

User or external training, backlog refinement, or backlog items also take place in this phase.

Overall, the tech companies work to create a beta version of the existing software. Then analyses the user acceptance, fix bugs then goes for the production for the final product version.

Phase 5: Implementation & Deployment

After testing & fixing all defined bugs, now it’s time to complete the development & ready to be deployed. In this phase, the operation team makes themselves busy getting the software live and running smoothly. To deploy it, the team would need to keep all of the things in mind:

- Getting all the software, servers, and hardware running for the final release.
- Setting up the databases and links to ensure all is ready.

After all, in the deployment phase, the vital round of quality assurance is mainly done. After that, the development team provides additional training to the users, define & resole new bugs to run the software smoothly.

Overall, a continuous flow of transparent iterations happens by upgrading & adding new features or functionalities in order to refresh the product.

Phase 6: Review & Retirement

This is the last phase of the Agile model of software development life cycle. Leave the maintaining responsibilities after deploying the software out of a perfect **Agile project management rule**.

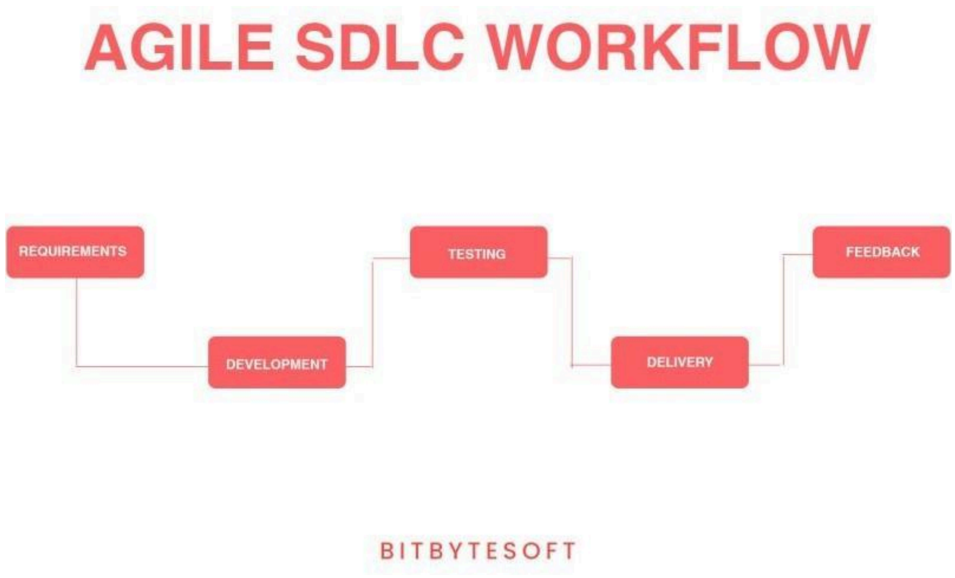
So, the development team runs the back-channel communication, takes the user feedback & shows the outcomes of the project to the clients.

It enhances usability is the leading role of this agile software development phase. When your built system is cloud-based (website, software, or app), the operation team will ensure the regular update of the software and hardware eligibility for taking the load.

But sometimes a software enters into the retirement stage due to replace with new applications or if the software has its own incompatibility over time.

Otherwise, take the client or stakeholder feedback to improve further & enhance customer satisfaction takes place, and Agile remains.

The Agile SDLC Workflow



The average release cycle of software consists of 1 to 4 weeks. According to the Agile software development phases, the developer team conducts an Agile workflow diagram. From start to the final completion of software development, the Agile iteration workflow consists of five steps-

Planning

Approach to software development and planning is the Agile workflow's first step. In the planning stage, the developer team imagined the whole project & made the sprint backlog, calculated the average time of release product, & determined about the required agile technologies.

Building

The building is the second step of a traditional Agile process model. In the building or design process, the Agile development team creates input functions & prepares the system for start testing.

Testing

Agile model in software development life cycle, this is the most vital step. In this step, the whole system is tested before the final production approach.

The quality assurance team works to find bugs and analyze the user flow diagram to fix and enhance the user acceptance of the application.

Delivery

The Agile delivery is the fourth step of the software development workflow. After performing a series of testing, user training, enhancing system functionalities developers team release the software.

Feedback

Review is the final step of the Agile development process. Client & user feedback is taken in this step to see the outcomes & if the software works and is functional or not.

All in all, this is the step of application retirement, either a continuous flow of review & up-gradation of the software.

Effective Agile Process For You Software Development

First thing first, how Agile software development life cycle works? Choosing perfect Agile methodologies does half of the work done in **software development**.

There have different Agile development life cycle models, including the traditional waterfall model, V-shape, Spiral Agile software development methods etc. Therefore, the Agile SDLC benefits depend on the methods and the project.

But, some standard processes require an effective Agile implementation, whichever method you choose.

Define Goals

Defining goals helps to get a clear view of the project objectives. In that case, the development team can determine what to do, how to do & what to avoid.

It also helps identify the requirements, work structure, and necessary tools and technologies for the projects.

Daily Meetings

Daily meetings improve communication between the clients and team. It’s a hybrid approach to open discussion, brainstorming, each development moving forward, and having an efficient Agile environment.

Interaction

Smooth interaction between design, development team & customer utilizes most Agile benefits. In other words, user interaction is the backbone of the Agile environment.

In order to best perform and achieve the internal goals supports between the team, open discussions are crucial to progress.

Collaboration

A collaborative environment between the development team and stakeholder ensures better workflow, empower between teams and finally bring out a high-quality product that can increase the business value.

Share Feedback

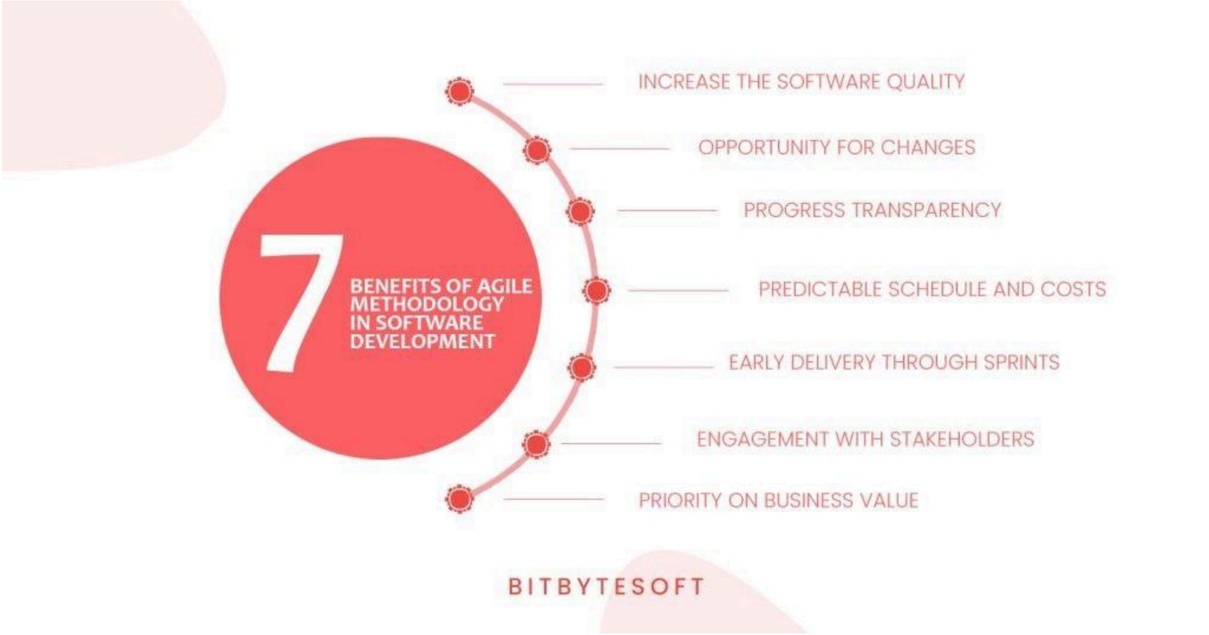
Clients and customer feedback are other core elements for successful software development. So, the development company or **Agile team** should receive feedback from time to time to upgrade features and present the existing product as a new one.

Remain Agile

You may remain run the same phases, which is the Agile software development benefits. If the software has compatibility & usability, you can run the same workflow to enhance the system's keep going & up-gradation.

Keep taking feedback and make changes as a continuous process of your software development life cycle.

Benefits of Agile Methodology in Software Development


7
BENEFITS OF AGILE METHODOLOGY IN SOFTWARE DEVELOPMENT
INCREASE THE SOFTWARE QUALITY
OPPORTUNITY FOR CHANGES
PROGRESS TRANSPARENCY
PREDICTABLE SCHEDULE AND COSTS
EARLY DELIVERY THROUGH SPRINTS
ENGAGEMENT WITH STAKEHOLDERS
PRIORITY ON BUSINESS VALUE
BITBYTESOFT

1. Increase the Software Quality

What is the biggest benefit of Agile development methodology? In the Agile method, every project can break down into multiple units that serve an easily manageable process. Enables the team on proper development and testing that reaches a high-quality assurance level.

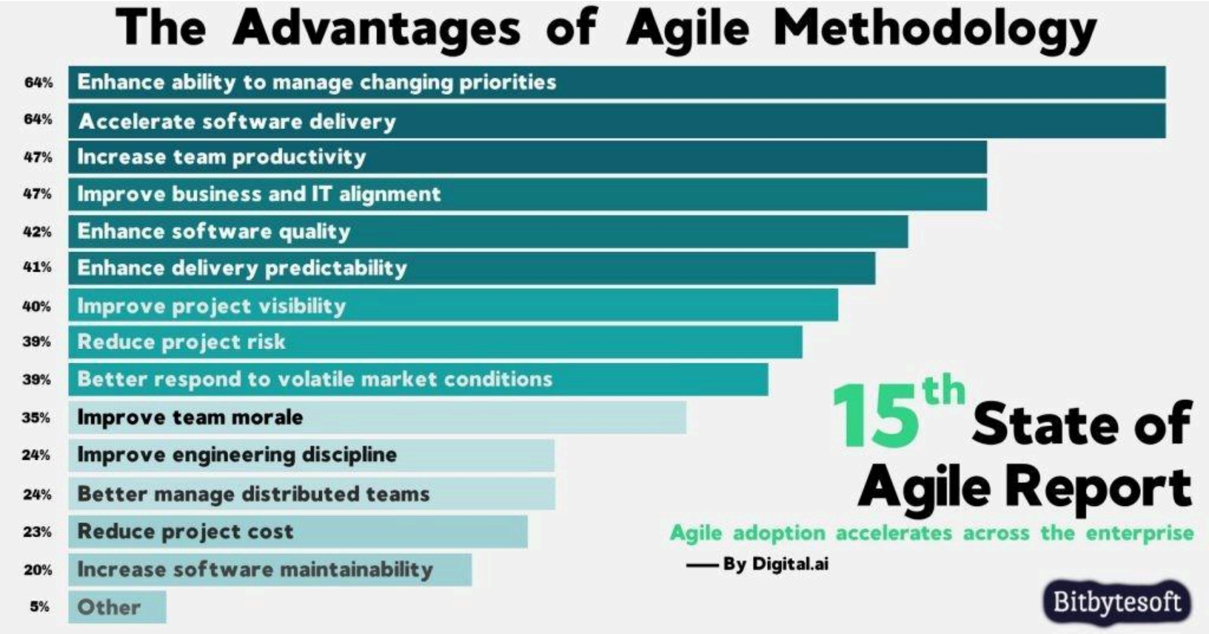
2. Opportunity for Changes

Every software company runs its system through **agile SDLC methodology**. Because Agile method gives teams a chance to time-to-time flexible change process and refines the complete product life cycle.

A GI LE	**WATER FALL**
The plan evolves over time	The plan is developed at the beginning
Iterative and incremental processes	Phased and sequential processes
Produces working outcomes on a regular basis	Delivers a final product at the end

A GI LE	WATER FALL
Cross-Functional Teams	Functional Teams

3. Progress Transparency



Source: Digital.ai

Clients or product owners can have enough involvement with their project by going an agile approach. They participate in the process, from planning software iterations, prioritizing functionalities, and reviewing sessions to regular software releases.

4. Predictable Schedule and Costs

Cost and schedule are fixed on the agile software development methodologies. This means every cost is predictable that can be executed within a specified time frame by the team. In this way, the client gets the cost of each feature and sprint plan.

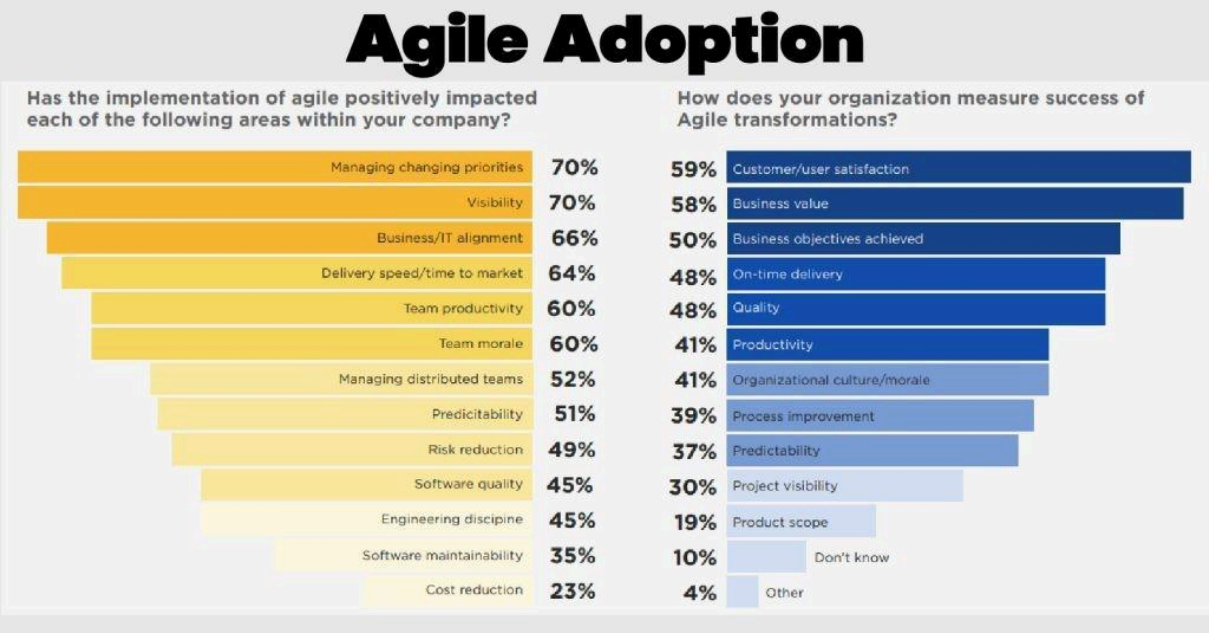
5. Early Delivery Through Sprints

An Agile sprint cycle lasts about 1 to 4 weeks, using schedule permits teams to serve additional features rapidly with a high level of unity. In addition, it gives them a chance to execute a software release or testing on the required software earlier than planned as there's sufficient business value.

6. Engagement with Stakeholders

The agile models offer several chances for team and client engagement on the entire project. And there is a much higher opportunity for collaboration between clients and the project team. Through frequent working software delivery, teams will get trusted by the client on their capability to come up with top-notch quality software. It makes the business stakeholders to be engaged in the process.

7. Priority on Business Value



Source: Digital.ai

The client is free to figure out the priority of features. Then the project manager & the team determines the most important thing, such as prioritizing the client's business goals. Therefore, they can provide features that create the most business value in clients' minds.