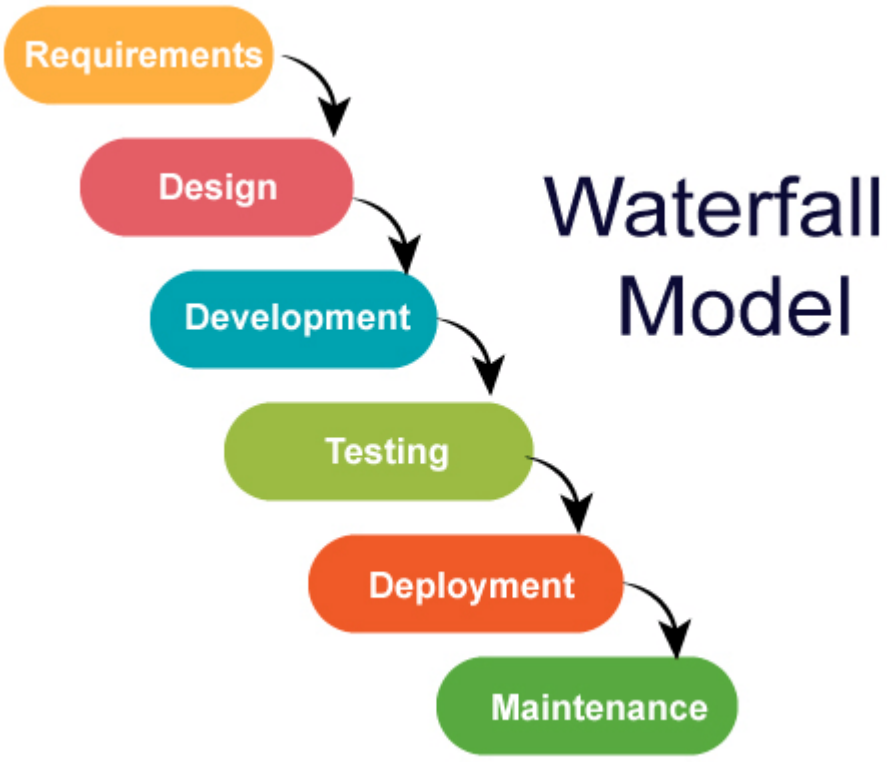
Assignment-2

Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of software development model is basically used for the project which is small and there are no uncertain requirements.

In this model software testing starts only after the development is complete. In waterfall model phases do not overlap.



Phases of Waterfall Model in Software Engineering

Requirements Gathering and Analysis

In this phase the requirements are gathered by the business analyst and they are analyzed by the team.
Requirements are documented during this phase and clarifications can be sought.
The Business Analysts document the requirement based on their discussion with the customer.
Going through the requirements and analyzing them has revealed that the project team needs answers to the following questions which were not covered in the requirements document –
Will the new banking application be used in more than one country?
Do we have to support multiple languages?
How many users are expected to use the application? etc

System Design

The architect and senior members of the team work on the software architecture, high level and low level design for the project.
It is decided that the banking application needs to have redundant backup and fail over capabilities such that system is accessible at all times.
The architect creates the Architecture diagrams and high level / low level design documents.

Implementation

The development team works on coding the project.
They take the design documents / artifacts and ensure that their solution follows the design finalized by the architect.
Since the application is a banking application and security was a high priority in the application requirements, they implement several security checks, audit logging features in the application.
They also perform several other activities like a senior developer reviewing the other developers code for any issues. Some developers perform static analysis of the code.

Testing

The testing team tests the complete application and identifies any defects in the application.
These defects are fixed by the developers and the testing team tests the fixes to ensure that the defect is fixed.
They also perform regression testing of the application to see if any new defects were introduced.
Testers with banking domain knowledge were also hired for the project so that they could test the application based on the domain perspective.
Security testing teams were assigned to test the security of the banking application.

Deployment

The team builds and installs the application on the servers which were procured for the banking application.
Some of the high level activities include installing the OS on the servers, installing security patches, hardening the servers, installing web servers and application servers, installing the database etc.
They also co-ordinate with network and IT administrative teams etc to finally get the application up and running on the production servers.

Maintenance

During the maintenance phase, the team ensures that the application is running smoothly on the servers without any downtime.

Advantages of waterfall model

1. This model is simple and easy to understand and use.
2. It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
3. In this model phases are processed and completed one at a time. Phases do not overlap.
4. Waterfall model works well for smaller projects where requirements are clearly defined and very well understood.

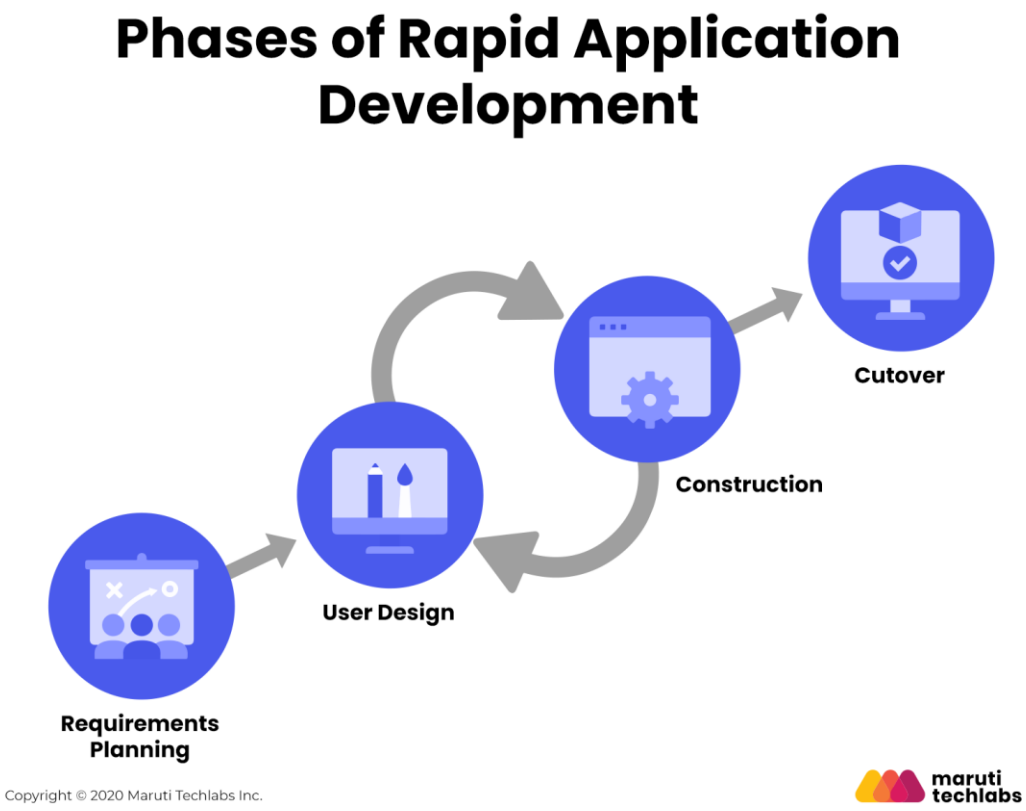
Disadvantages of waterfall model

1. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
2. No working software is produced until late during the life cycle.
3. High amounts of risk and uncertainty.
4. Not a good model for complex and object-oriented projects.
5. Poor model for long and ongoing projects.
6. Not suitable for the projects where requirements are at a moderate to high risk of changing.

Rapid Application Development

Rapid application development (RAD) is an agile project management strategy popular in software development.The key benefit of a RAD approach is fast project turnaround, making it an attractive choice for developers working in a fast-paced environment like software development. This rapid pace is made possible by RAD's focus on minimizing the planning stage and maximizing prototype development.By reducing planning time and emphasizing prototype iterations, RAD allows project managers and stakeholders to accurately measure progress and communicate in real time on evolving issues or changes. This results in greater efficiency, faster development, and effective communication.

You can break down the process in a few ways, but in general, RAD follows four main phases.



Phase 1: Requirements planning:-

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project.
During this stage, developers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:
1. Researching the current problem
2. Defining the requirements for the project
3. Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and developer, teams can avoid miscommunications and costly change orders down the road.

Phase 2: User design:-

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.
This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with developers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.
All the bugs and kinks are worked out in an iterative process. The developer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.
This method gives developers the opportunity to tweak the model as they go until they reach a satisfactory design.
Both the software developers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

Phase 3: Rapid construction:-

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.
Because the majority of the problems and changes were addressed during the thorough iterative design phase, developers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

1. Preparation for rapid construction
2. Program and application development
3. Coding
4. Unit, integration, and system testing

The software development team of programmers, coders, testers, and developers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.
This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

Phase 4: Cutover:-

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training.

All final changes are made while the coders and clients continue to look for bugs in the system.

Advantages of the RAD model:

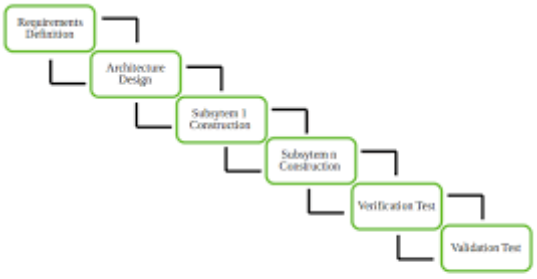
1. Reduced development time.
2. Increases reusability of components
3. Quick initial reviews occur
4. Encourages customer feedback
5. Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model:

1. Depends on strong team and individual performances for identifying business requirements.
2. Only system that can be modularized can be built using RAD
3. Requires highly skilled developers/designers.
4. High dependency on modeling skills
5. Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

Incremental Model

Incremental model is the software development process in which the requirements would be broken down into number of standalone increments of the software development cycle.



Every increment is treated as a sub project and that follows all the phases of the SDLC life cycle. Incremental development follows all the phases from Requirements, Analysis, Design, Development, Testing and Implementation. The developing functionality in each increment is an addition to the previously developed functionality. These increments cycles gets repeated until the targeted software is fully developed.

Every increment is assigned with a priority and high priority requirements of the software handled first. Once the increment is developed, then the specific increment gets freezed and moves to the next increment requirements.

At any specified time, the plan focuses on the current increment only without having any kind of long term plans. At each increment of the incremental model, there would be a thorough review to decide whether to continue to the next phase or not.If the increment passed the testing or validation, them moves to the next increment in the queue as per priority.

Incremental model is preferred because of the way it divides the software development into sub increments and each sub increment is further developed by completing all the phases of SDLC successfully. By implementing this model, we can make sure that we are not missing any objectives that are expected from final software product.

We can also assure that the final software is defect free and also every increment is compatible with the previously developed product and future developing product.

Requirement Phase

The incremental model in software engineering begins with the requirement phase. This phase is extremely important in product development because developing a solution that gives value to clients is impossible without understanding the initial requirements. At this stage, business analysts and project managers gather functional requirements as well as non-functional requirements from potential clients.

Having completed the requirement analysis process, the team should collect all of the project's requirements in one document. After gathering requirements, the Business Analyst (BA) converts them into technical requirements with the help of senior team members such as the Subject Matter Expert (SME) and the team leader. A document called the SRS (Software Requirement Specification) contains detailed specifications of the technical requirements. After the SRS has been created, we need to send it to the client or to SMEs on the client's side for approval before proceeding with the next phase. The SRS will be a reference document for the next phase.

Design and Development Phase

During this phase, the product architect develops an optimized product architecture by using the SRS document. The product architect creates a rough design, working models, specifies how the software works, how the new design looks, how the control flows from one screen to another, etc., based on the requirements provided by SRS. When they are done creating the product architecture, they will save it in a DDS (Design Document Specification). Diagrams like Entity-Relationship Diagrams (ERDs) and Data Flow Diagrams (DFDs) may be included in this document.

Once approved by all stakeholders, the DDS document can be implemented. Eventually, overall system architecture is designed by defining each module's functionality (capability) and its interaction with other modules or cross systems.

Next, developers begin development by following the coding standards established by the organization. Code is developed from scratch according to the requirements in SRS, as well as the design specifications in DDS.

Developing clean and efficient code can have a significant impact on the performance of software, so programmers must write code in an organized and detailed manner. As well, the language used to write the software code can vary depending on the type of software being created, its objectives, and its environment of use. Upon completion of coding, developers use the programming tools to compile and debug the new code to ensure that it works properly.

Testing Phase

In this phase, once the code is written, it is tested to determine whether it works as expected. Prior to handing over code to the testing team, the developer performs initial testing such as unit testing and/or application integration testing. If all goes well, the code is moved to the testing environment.

From there, the testing team will perform the testing. There are several types of testing the testing team performs quality assurance (QA) testing, system integration testing (SIT), user acceptance testing (UAT), and approval testing. Testing is done to determine if the code and programming meet customer/business requirements. Before the implementation phase begins, companies can identify all bugs and errors in their software during the testing phase. Software bugs can jeopardize (put someone/something in a risky position) a client's business if they aren't fixed before deployment.

Implementation Phase

The incremental model of software development has reached its final phase. The product is ready to go-live after it has been tested and has passed each testing phase. The product is therefore ready to be used in a real-world environment by end users.
After the software is fully tested and is free of errors and defects, the client reviews the test results and approves the deployment. Upon product deployment, the new functionality becomes available to the current end-users of the system. In some cases, product deployment may be divided into several phases based on the company's business strategy. A user acceptance test is one of the ways companies seek feedback from the intended users. Thus, the software can be tested in a pre-production environment, before a real-world production, thereby leading to better results.

Advantages of Incremental model:

1. Generates working software quickly and early during the software life cycle.
2. This model is more flexible – less costly to change scope and requirements.
3. It is easier to test and debug during a smaller iteration.
4. In this model customer can respond to each built.

5. Lowers initial delivery cost.
6. Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages of Incremental model:

1. Needs good planning and design.
2. Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
3. Total cost is higher than waterfall.

Spiral Model

Spiral Model is a risk-driven software development process model. It is a combination of waterfall model and iterative model. Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process.
Each phase of spiral model in software engineering begins with a design goal and ends with the client reviewing the progress.
.
The development process in Spiral model in SDLC, starts with a small set of requirement and goes through each development phase for those set of requirements. The software engineering team adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase. The below figure very well explain Spiral Model:

Spiral model phases

The different phases of spiral model are-

Planning

This phase begins by gathering business requirements into a baseline spiral. In the subsequent spiral as the product matures, all system, subsystem and unit requirements are identified at this stage.
This phase also includes understanding system requirements through ongoing communication between the customer and system analysts. At the end of the spiral, the product will be deployed in the identified market. This includes iteration cost, schedule, and resource estimates. This includes understanding system requirements for ongoing communication between system analysts and customers.

Risk Analysis

After the “plan” phase, the team prepares for the “risk” phase. The “risk” phase is designed to take into account the variability in the rate at which a given product might fail. It is designed to account for the uncertainty in the rate at which a given product might fail. During the “risk” phase, the team evaluates various aspects of the current state of the product, such as the state of its code, the state of its design, and the state of its prototype. The team then makes adjustments to the current state of the product based on the changes made in the “plan” phase, and then follows up with a “sales” phase to collect customer feedback.
Once risks are identified, risk mitigation strategies are planned and completed.
Briefly, risk analysis involves identifying, estimating and monitoring technical feasibility and management risks such as: schedule slippage and cost overrun. After testing the build, at the end of the first iteration, customers rate the software and provide feedback.

Product development

In the next quadrant, prototypes are built and tested. This step includes architectural design, module design, physical product design and final design. Convert the proposals made in the first two quadrants into usable software.
This phase also includes the actual implementation of features in a project which are verified by performing testing.

Next phase planning

In this phase ,the software is evaluated by the customer and feedback is given. The team prepares for the next phase of the planning process. The next phase of the planning process is known as the “spiral” phase. During the “spiral” phase, the team determines the order of events in the current state of the product and then follows these events up with a “revision” phase to “Revise” the current state of the product so that it is ready for production. The “revision” phase is also called the “reproduction” phase, and it is one of the most important aspects of the planning process.

Benefits /Advantages of the spiral model

The benefits of the spiral model include:

1. The spiral model works for development as well as enhancement projects.
2. The spiral model is a software development model designed to control risk.
3. The major distinguishing feature of the spiral model is that it creates a risk-driven approach to the software process rather than a primarily document-driven or code-driven process.
4. It incorporates many of the strengths of other models and resolves many of their difficulties.

5. Spiral models are ideal for large and complex projects as continuous prototyping and evaluation helps reduce risk.

6. This model supports customer feedback and the implementation of Change Requests (CR), which is not possible with traditional models such as waterfall.

7. Customers see the prototype at each stage, which increases the chances of customer satisfaction.

Limitations of the spiral model

1. The most obvious limitations of the spiral model are that it does not account for the uncertainty in the rate at which a given product might fail, and for the variability in the rate at which a given product might fail.
2. This model does not account for the fact that customers might change their mind about a given product and return it to the supplier.
3. Spirals can continue indefinitely. As the spirals continue to increase,the cost of project also increase.

4. Many intermediate stages require excessive documentation.
5. Spiral models are best suited for projects rather than large complex small projects as continuous prototyping and evaluation helps reduce risk.
6. The number of phases is unknown at first, and frequent prototyping and risk analysis can exacerbate the situation, so project deadlines may not be met.
7. This is very expensive and time consuming as each stage involves prototyping and risk analysis.

Agile model

In the Agile process model, each iteration is a small-time "frame" that lasts anywhere from one to four weeks. This time frame is termed as Time Box i.e. the amount of time taken to finish an iteration. The greatest length of time required to provide an iteration to clients is referred to as a time-box. As a result, the end date for an iteration remains unchanged. Though, if necessary, the development team might choose to limit the delivered functionality during a Time-box in order to meet the deadline.

The delivery of an increment to the client after each Time-box is the basic principle of the Agile approach.

The segmentation of the complete project into smaller pieces helps to reduce the total project delivery time requirements while minimizing project risk. Before a working product is demonstrated to the client, each iteration requires a team to go through the entire software development life cycle, which includes planning, requirements analysis, design, coding, and testing.

The four essential values of agile software development are highlighted:

- ✓ Interactions between individuals and groups about processes and tools
- ✓ Working software takes precedence over thorough documentation.
- ✓ Collaboration with customers is preferred over contract negotiations.
- ✓ Adapting to change in accordance with a strategy

Phases of Agile Model

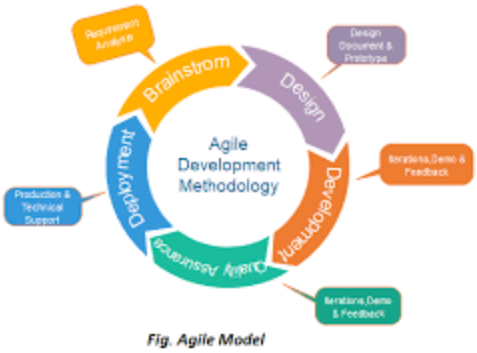


Fig. Agile Model

Different phases of Agile Methodology

Requirement Gathering:

You must define the criteria at this step. You should describe the project's business opportunities and estimate the time and effort required to complete it. You can assess technical and economic feasibility based on this information.

Design the requirement:

Work with stakeholders to define requirements once you've defined the project. To demonstrate how new features function and how they will fit into your existing system, use a user flow diagram or a high-level UML diagram.

Develop/Iteration:

The work begins once the team has defined the requirements. Designers and developers begin work on their projects, with the goal of releasing a functional product. The product will go through several stages of development before being released, thus it will have basic, rudimentary functionality. Ultimately, deploying a non-static product or service.

Test:

This phase basically involves the testing team i.e. the Quality Assurance team checks the product's performance and seeking for the bug during this phase.

Deployment:

The team creates a product for the user's work environment in this phase.
Review / Feedback: The final phase is to get feedback after the product has been released. This is where the team receives feedback on the product and works through it.

Advantages of Agile model:

1. Customer satisfaction by rapid, continuous delivery of useful software.
2. People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
3. Working software is delivered frequently (weeks rather than months).
4. Face-to-face conversation is the best form of communication.
5. Close, daily cooperation between business people and developers.
6. Continuous attention to technical excellence and good design.
7. Regular adaptation to changing circumstances.
8. Even late changes in requirements are welcomed

Disadvantages of Agile model:

1. In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
2. There is lack of emphasis on necessary designing and documentation.
3. The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
4. Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

V Model

The V model is an SDLC model in which process execution occurs sequentially in a V shape. This is also called the validation and validation model.

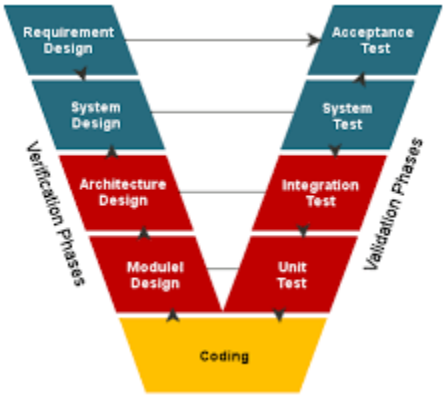
The V model is an extension of the waterfall model and is based on assigning test phases to corresponding development stages. This means that every development cycle has a directly linked testing phase.

This is a highly disciplined model; the next phase will only start after the previous one is completed.

V model – design is a method for structuring and planning computer software development. The first step in implementing an idea in software engineering is to model the system conceptually. Conceptual modeling involves creating models using diagrams, tables, flowcharts, or other means to accurately and thoroughly depict a system.

Before starting development, the model should be clear, concise, and easy to understand. After conceptually modeling the system, developers start developing the source code for the application.

During this phase, developers must test the source code thoroughly to avoid making any mistakes that would negatively affect the end product.



Here you can see that each development phase is connected to a test phase. The development and testing happen in parallel, as shown in the diagram.

The left half of the V represents validation, the right half represents validation, and both halves are connected by a coding phase that gives the V.

During the validation phase, static analysis is performed. Check if the current phase meets the desired requirements without executing the code. During the validation phase, dynamic analysis is performed. Verify whether the current phase meets the desired requirements the customer makes for the software by running the code.

V-model phases

Verification phase:

The primary purpose of the verification process is to ensure the quality of software applications, designs, architectures, etc. Validation processes include activities such as reviews, walkthroughs, and inspections. Are you building the right product? This line is used for Verification. After reading it, you can understand that this line is asked during the project development. So basically, Verification is done when the process is in the development phase and is not complete.

So Verification is done after every phase of project development.

Business Requirements Analysis

In this phase, customer requirements and needs are understood. What the customer expects from the final software and what features the customer wants are all discussed during this phase. This is a very important stage as there often needs to be more clarity between both customers and developers regarding the end result of the software. In this phase, acceptance tests are performed.

System Design

This phase determines the actual design of the system. After the requirements analysis phase, the full design of the system is considered based on the final requirements. This includes hardware and communication setup requirements.

Architectural Design

This phase is also called High-Level Design (HLD). After the system design is analyzed, the architecture of the system is determined. It consists of various modules, database tables, UML diagrams, etc. All communication between the system's internal modules and external systems is understood during this phase.

Modular Design

This phase is also called Low-Level Design (LLD). After analyzing the high-level design, each component of the high-level design is discussed in detail. The system is broken down into small modules in the module design phase. The compatibility of individual internal modules and their feasibility is checked. During this phase, unit tests are executed.

Implementation and testing

After the implementation phase is complete, the team tests their software. If testing reveals errors in the software, developers must be able to identify and resolve these issues quickly so that users can safely interact with their software. Once bugs are fixed, Verification begins- a process in which independent parties test the software to ensure it functions as advertised.

Validation phase:

Validation in software engineering is a dynamic mechanism for verifying that a software product meets the customer's requirements accurately. This process helps ensure that the software provides the desired benefits in the right environment.

The validation process includes integration testing, unit testing, system testing, and user acceptance testing.

1. Unit Testing: In unit testing unit Test Plans (UTPs) are created during the module design stage in the V model.
2. Integration Testing: In integration testing an integration test plan is created during the architecture design stage.
3. System Test: In system testing, system test plan is created during the design stage.
4. Acceptance Testing: In acceptance testing refers to the business requirements analysis part.

Advantages of V model

1. This is a very professional model for developing software.
2. Covers all functional areas.
3. It contains instructions and recommendations that provide a detailed description of the issue encountered.
4. Emphasize the importance of testing and ensure testing is planned. There are significant changes to catching bugs in the early stages.
5. Test planning and design are performed during this phase as pre-coding testing activities.

Disadvantages of V model

1. V-model as being too rigid or prescriptive when applied to real-world situations.- For example, some critics point out that no project is ever completed according to an exact timeline when actual software development occurs
2. The risks are high and unpredictable.
3. This model is not suitable for complex projects.
4. After completing the testing phase, it isn't easy to go back to the previous phase.
5. This is a poor model for long-running, ongoing projects.

Comparison between the models

S.NO	MODELS	STRENGTH	WEAKNESS	TYPES OF PROJECT
1	Waterfall Model	Simple. Easy to execute. Intuitive & logical.	Simple approach. Do not allow changes. Cycle time is too long. User feedback is not allowed.	For well-understood problem. Short duration project. Automation of adjusting manual system.
2	Prototyping Model	Helps in requirement elicitation. Reduce the risk. Lead to a better system.	Possibly higher cost. Do not allow labour charges.	System With massive users. Uncertainty in requirement.
3	Iterative Enhancement Model	Regular/Quick delivery. Reduces risk. accumulate Changes. prioritizes requirement.	Each Iteration Can Have Planning Over Head. Cost may increase as work done in one iteration may have to be undone later. System design & structure may suffer as frequent changes are made.	For a business where time is of the essence. Where the risk of the long project can be taken.
4.	Spiral Model	Control project risk. Very flexible. Less document needed.	No strict standard for software development. No particular began and ending of the particular phase	Project build on untested assumptions
5.	V Model (Validation and Verification model)	Emphasize the importance of testing and ensure testing is planned Test planning and	This model is not suitable for complex projects. After completing the testing phase, it isn't	V-Model can be used on small- to medium-sized software projects, where the requirements are clearly defined and fixed

		design are performed during this phase as pre-coding testing activities	easy to go back to the previous phase.	and not ambiguous.
6.	Agile Model	Working software is delivered frequently Regular adaptation to changing circumstances Even late changes in requirements are welcomed	There is lack of emphasis on necessary designing and documentation. The project can easily get taken off track if the customer representative is not clear what final outcome that they want	When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.

When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.