RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

II Year-III Semester: B.Tech. Computer Science and Engineering (Data Science)

3CDS4-07: Software Engineering

Credit-3
3L+0T+0P

Max. Marks : 100 (IA:30, ETE:70)
End Term Exam: 3 Hours

CS Section-C 2103 3CS4-07

SN	CONTENTS	Hours
1	Introduction, software life-cycle models, software requirements specification, formal requirements specification, verification and validation.	8
2	Software Project Management: Objectives, Resources and their estimation, LOC and FP estimation, effort estimation, COCOMO estimation model, risk analysis, software project scheduling.	8
3	Requirement Analysis: Requirement analysis tasks, Analysis principles. Software prototyping and specification data dictionary, Finite State Machine (FSM) models. Structured Analysis: Data and control flow diagrams, control and process specification behavioral modeling	8
4	Software Design: Design fundamentals, Effective modular design: Data architectural and procedural design, design documentation.	8
5	Object Oriented Analysis: Object oriented Analysis Modeling, Data modeling. Object Oriented Design: OOD concepts, Class and object relationships, object modularization, Introduction to Unified Modeling Language	8
	TOTAL	**40**

Office of Dean Academic Affairs
Rajasthan Technical University, Kota

Unit 1 : Software Engineering

⇒ Software Engineering

To understand the concept of Software Engineering first we have to understand the meaning of Software & Engineering.

Software: Software is a program or set of programs containg instructions that provide desired functionality.

Engineering: is the process of desining and building something that serves a particular purpose & find a cost effective solution to problems

★ **Software Engineering:** is the process of desiging, developing, testing and maintaing software. it is a systematic and disciplined approach to for software development, that aims to create high quality, reliable and maintainable Software

Software Engineering includes a variety of techniques, tools & methodologies including requirement analysis design, testing & maintainance.

⇒ Why Software Engineering required?

- To manage large Softwares
- For more scalability
- Cost management
- Dynamic Nature
- Quality management

→ The necessity of Software engineering appears to higher state of progress in user requirement or a Program is working.

~~Huge Programming~~:

Importance/benifits of Software Engineering

There are several advantages to using a systematic disciplined approach for Software development such as.

- → improved Quality -
- → Reduce complexity -
- → To minimize Software cost -
- → To decrease time -
- → Reliable Software -
- → increased Customer Satisfication -
- → Handaling big projects
- → effectiveness:

→ Reduce Complexity:- Big Softwares are always complex and challenging to progress. Software Engineering has a great solution to reduce the complications of any project. S divides big problems into small issues and then start solving each issue one by one. all problems solved one by one indepe -ently to each other.

→ To minimize Software Cost: By identifying and addressi potential problems early in the development process, SE can help to reduce the cost of fixing bugs & adding new features l ch.

→ improved quality: By following established Software engineering principles & techniques, the Software can be developed with fewer bugs & higher reliability.

→ Customer Satisfication: By involving customer in the development ~~Stats~~ process & developing Software that meet their needs, Software Engineering can help to increase customer satisfaction.

→ To decrease time: anything that is not made according to the proper always waste time. everything should be well maintained.

→ Reliable S/W:- Software should be secure, means if you delivered the S/W, then it should work for at least given time or subscription. If any bugs come then company is responsible

For saving all bugs. Bcoz in SE testing & maintance is given.

Effectiveness: Effectiveness come if anything has made according to the standards. S/w standards are big target of companies to make it more effective. So S/w become more effective on the act with the help of SE.

Disadvantages:
- High upfront cost
- Complexity
- High learning curve
- High dependence on tools
- High maintance.

→ High upfront cost: implementing a systematic approach to s/w development can be resource intensive & require a significant investment in tools & training

→ Complexity: with increase no of tools & methodology SE can be complex & difficult to navigate.

→ High learning curve: The development process can be complex and it requires a lot of learning & training which can be challenging for new developers.

→ High dependence on tools: SE heavily depends on tools. & if tools are not properly configured or are not compatible with s/w then it may cause issues.

→ High maintance: The SE process requires regular maintance to ensure that the s/w is running efficiently, which can be costly & time consuming.

SDLC: Software Development Life Cycle.

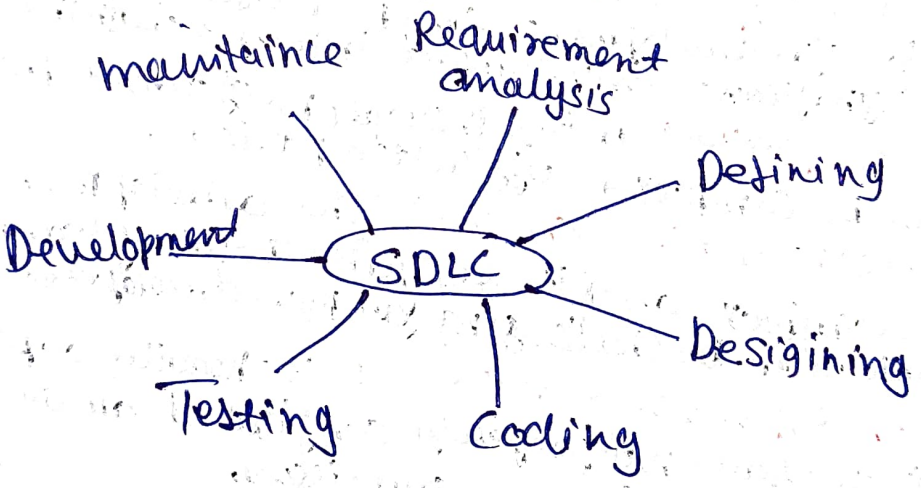
SDLC is a structured process that enables the production of high quality, low cost s/w, in the shortest possible production time. The goal of the SDLC is to produce superior s/w that meets and exceeds all customer expectation and demands.

Why SDLC important?

→ It provide a standardized framework that defines a deliverables.
→ It helps in project planning, estimating & Scheduling
→ It make project tracking & controlling easier.
→ It increase speed of development.
→ It improve client relations.
→ It decrease project risks.
→ It decrease project management expensis & the overall cost of production.
→ It increase visibility on all aspects of the life cycle to all stakeholders involved in development process.

→ SDLC cycle

DDLC cycle represents the process of developing S/w. The following steps are included in SDLC frame work.



Stage 1: Planning & Requirement analysis :-

Requirement analysis is one of the most important & fundamental stage in SDLC. It was performed by senior members of the team with inputs/ requirements from the customer the sales department, market survey & domains experts in the industry. This information is then used to ~~plan~~ plan the basic project approach & to conduct product feasibility Study in the economical, operational & technical area.

Planning for the quality assurance requirement and identification of the risk associated with the

Boject is also done in the planing stage.

Stage2: Defining Requirments:- once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer. this is done through SRS (S/W requirement specification) document which consist of all the product requirements to be designed & developed during the project life cycle.

Stage3: Desiging Boject architecture:-

SRS is the reference for product ~~architecture~~ architects to come out with the best architecture for the product to be developed. Based on the requirements ~~set~~ specified in SRS, usally more than one design approach for the product architecture is proposed and documented in DDS - Design document specification.

This DDS is viewed by all the important stake holders & based on various parameters as risk assesments, Products robustness, design modularity, budget & time constraints & then best design is selected for project.

Stage4: Building or developing product:-

In this stage of SDLC the actual development starts and the product built. The programme code is generated as per DDS during this stage.

Stage5: Testing the Product: This Stage is usually a subset of all the stages as in the modern SDLC models, the testing activites are mostly involved in all the stages of SDLC. In this stage defects are ~~testing~~ reported to the stage of ~~production~~ product where ~~product defects are reported, tracked, fixed~~ and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment and maintance.

once the product is tested and ready to be deployed it is released formally in the appropriate market, Sometimes product deployment happens in stages as per business strategy of that organization. the product may first released in a limited segments and tested in the real business enviroment.

Then based on the feedback, the product is released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintence is done for the existing customale based.

SDLC models

There are various s/w development life cycle models defined and designed which are followed during the s/w development process. These models are also referred as s/w development process models. each process model follows a series of steps unique to its type to ensure success in the process of s/w development

Some SDLC models are

- → waterfall model
- → iterative model
- → Spiral model
- → V model
- → Big Bang model.

other models are like Agile model, RAD model, Rapid application development & prototyping models.

Waterfall Model

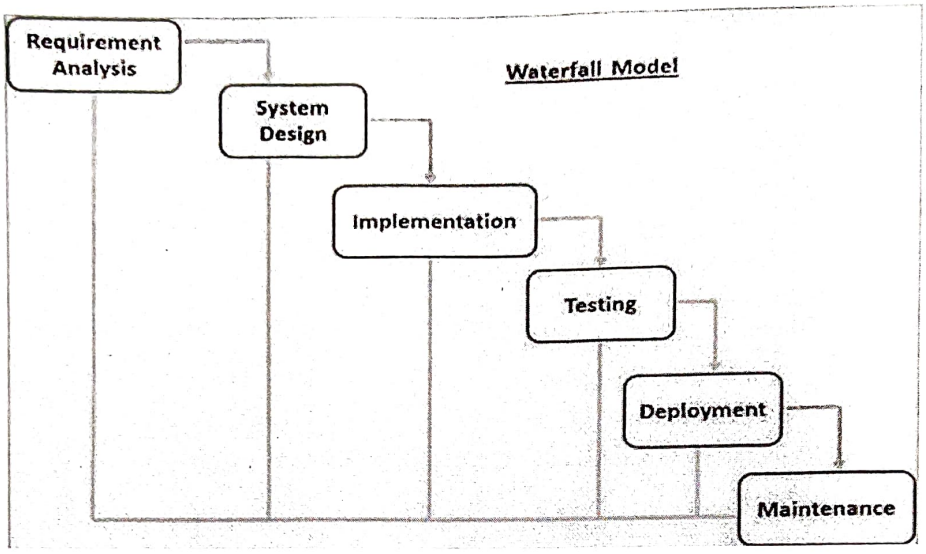
The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are –

- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Waterfall Model - Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are –

- Requirements are very well documented, clear and fixed.

- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

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

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows –

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

Iterative Model

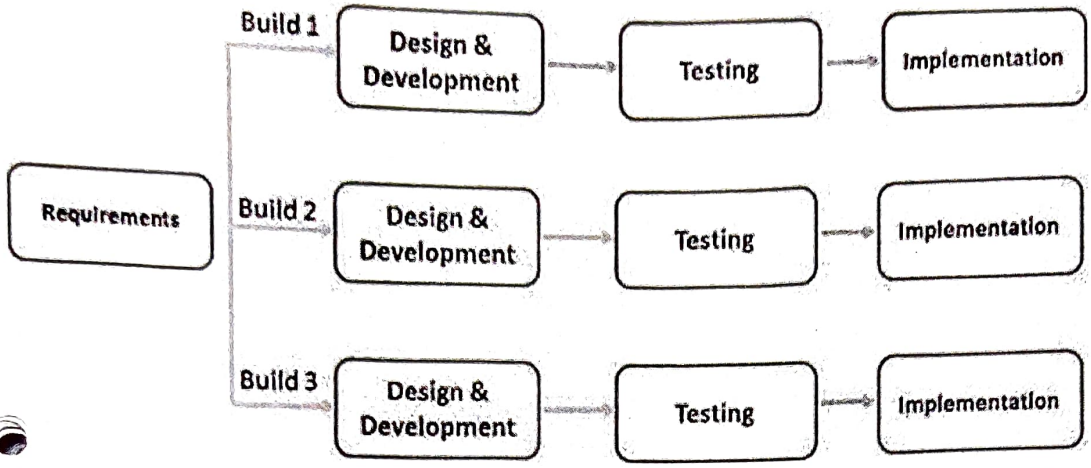
Inthe Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each **iteration of the model.**

Iterative Model - Design

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

The following illustration is a representation of the Iterative and Incremental model –



Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

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

The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software.

Iterative Model - Application

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

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- There are some high-risk features and goals which may change in the future.

Iterative Model - Pros and Cons

The advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

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

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During the life cycle, software is produced early which facilitates customer evaluation and feedback.

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

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

Spiral Model

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

Spiral Model - Design

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

Identification

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

gn

he Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

Construct or Build

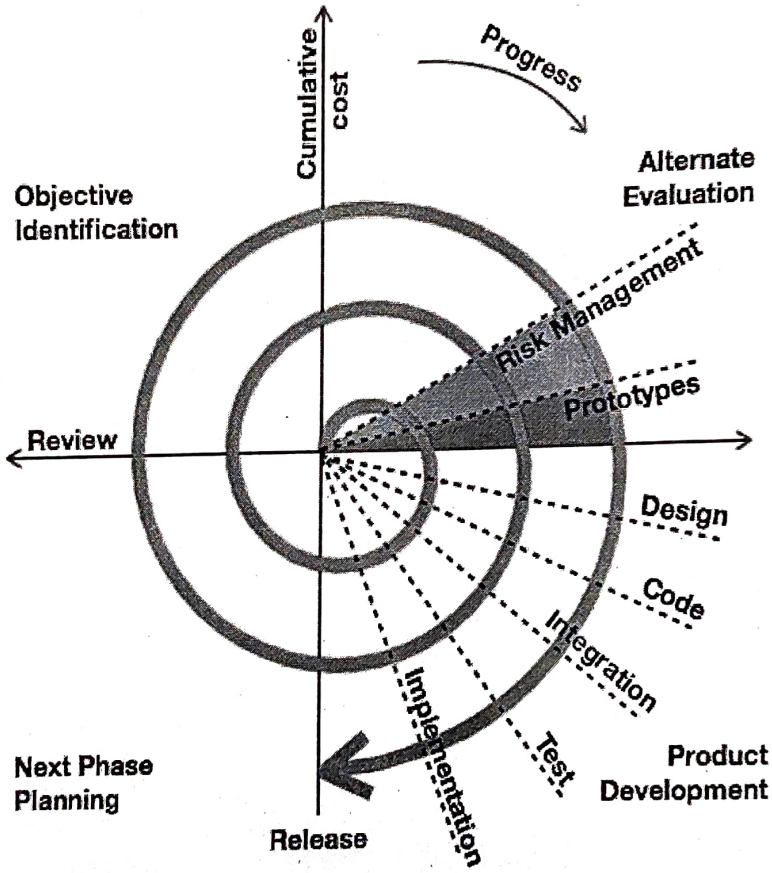
The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

Evaluation and Risk Analysis

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

The following illustration is a representation of the Spiral Model, listing the activities in each phase.



Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

Spiral Model Application

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model –

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Spiral Model - Pros and Cons

The advantage of spiral lifecycle model is that it allows elements of the product to be added in, when they become available or known. This assures that there is no conflict with previous requirements and design.

This method is consistent with approaches that have multiple software builds and releases which allows making an orderly transition to a maintenance activity. Another positive aspect of this method is that the spiral model forces an early user involvement in the system development effort.

On the other side, it takes a very strict management to complete such products and there is a risk of running the spiral in an indefinite loop. So, the discipline of change and the extent of taking change requests is very important to develop and deploy the product successfully.

The advantages of the Spiral SDLC Model are as follows –

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

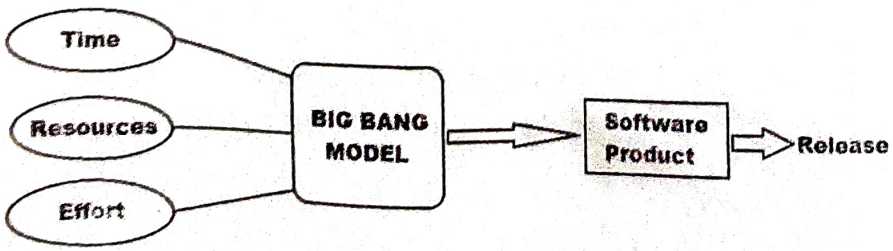
The disadvantages of the Spiral SDLC Model are as follows –

- Management is more complex.
- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.

Big Bang Model

The Big Bang model is an SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement. This Big Bang Model does not follow a process/procedure and there is a very little planning required. Even the customer is not sure about what exactly he wants and the requirements are implemented on the fly without much analysis.

Usually this model is followed for small projects where the development teams are very small.



Design :

The product requirements are understood and implemented as they arrive. The complete modules or at least the part of the modules are integrated and tested. All the modules are run separately and the defective ones are removed to find the cause. It is a suitable model where requirements are not well understood and the final release date is not given. In simple, it can be phased out in 3 points i.e.

1. Integrate each individual's modules to give a unique integrated overview
2. Test each module separately to identify any error or defects
3. If any error found then separate that module and identify the cause of the error

When to use it and where not to :

This SDLC model is suitable for small projects when few people are working on the project, the customer's demands are not exact and keep changing, or if it is a dummy/side-project. As there is no proper planning in this model it is considered the worst SDLC model and is highly unsuitable for large projects.

It is recommended to go for the Big Bang model only due to the following cases i.e.

1. Developing a project for learning purposes or experiment purposes.
2. No clarity on the requirements from the user side.
3. When newer requirements need to be implemented immediately.
4. Changing requirements based on the current developing product outcome.
5. No strict guideline on product release or delivery date.

Features of Big Bang Model :

- Not require a well-documented requirement specification
- Provides a quick overview of the prototype
- Needs little effort and the idea of implementation
- Allows merging of newer technologies to see the changes and adaptability

Pros of Big Bang Model :

- There is no planning required for this.
- Suitable for small projects
- Very few resources are required.
- As there is no proper planning hence it does not require managerial staffs
- Easy to implement
- It develops the skills of the newcomers
- Very much flexible for the developers working on it

Cons of Big Bang Model :

- Not suitable for large projects.
- Highly risky model and uncertain
- Might be expensive if requirements are not clear
- Poor model for ongoing projects

V-Model

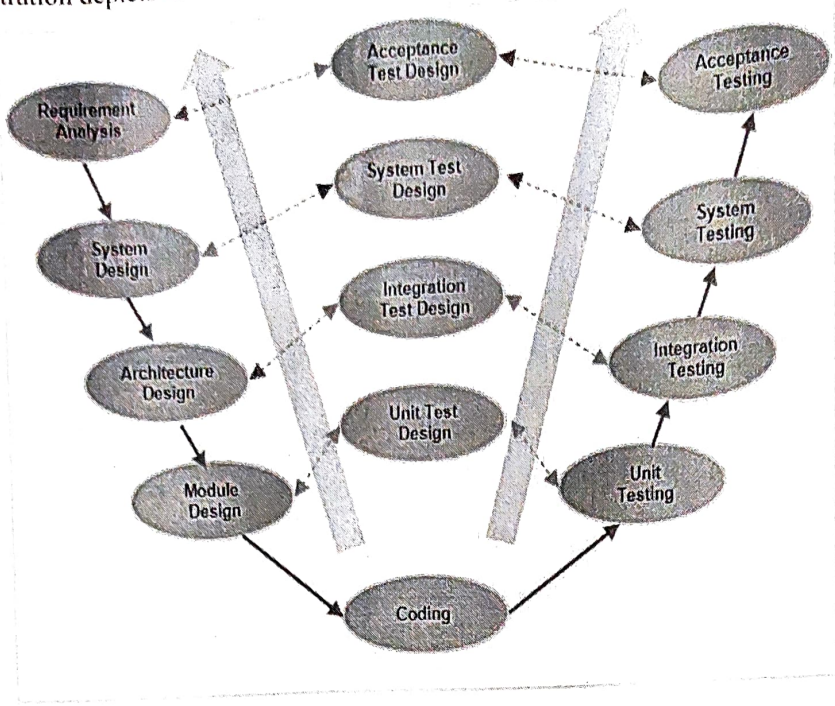
The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

V-Model - Design

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The following illustration depicts the different phases in a V-Model of the SDLC.



V-Model - Verification Phases

There are several Verification phases in the V-Model, each of these are explained in detail below.

Business Requirement Analysis

This is the first phase in the development cycle where the product requirements are understood from the customer's perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The **acceptance test design planning** is done at this stage as business requirements can be used as an input for acceptance testing.

System Design

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

Architectural Design

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as **High Level Design (HLD)**.

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

Module Design

In this phase, the detailed internal design for all the system modules is specified, referred to as **Low Level Design (LLD)**. It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

Coding Phase

actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements.

The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

Validation Phases

The different Validation Phases in a V-Model are explained in detail below.

Unit Testing

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

Integration Testing

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

System Testing

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

V- Model — Application

V- Model application is almost the same as the waterfall model, as both the models are of sequential type. Requirements have to be very clear before the project starts, because it is usually expensive to go back and make changes. This model is used in the medical development field, as it is strictly a disciplined domain.

The following pointers are some of the most suitable scenarios to use the V-Model application.

- Requirements are well defined, clearly documented and fixed.
- Product definition is stable.
- Technology is not dynamic and is well understood by the project team.
- There are no ambiguous or undefined requirements.
- The project is short.

V-Model - Pros and Cons

The advantage of the V-Model method is that it is very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today's dynamic world, it becomes very expensive to make the change.

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

- This is a highly-disciplined model and Phases are completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

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

- High risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- Once an application is in the testing stage, it is difficult to go back and change a functionality
- No working software is produced until late during the life cycle.

SRS Software Requirement Specification.

A S/W Requirement Specification (SRS) is a document which is used as a communication medium b/w the customers. A SRS is the most basic form of a formal document used in communicating the S/W Requirement b/w customer & developer

An SRS document concentrates on what need to be done and how to do. it is a complete discription of what customer actuly want.

→ Purpose of SRS

SRS is a comunication tool b/w customer and Business analyst, system developer.. maintance team, it can also be a contract b/w purchaser & supplier.

- → it will give firm foundation for the design phase
- → support project management & control
- → Help in controlling & evoluting of system.

→ Characteristics of SRS

- → correctness
- → Completness
- → Consistency
- → unabiguouness
- → Ranking for importance & stability
- → modifiability
- → verifiability
- → Traceability
- → Design independence
- → Testability
- → Customer Statisfication
- → The right level of abstraction

→ Properties of a good SRS document.

The essential properties of a good SRS document also of

→ Concise: The SRS report should be concise and at the sam

, unambiguous, consistent and complete. ~~verbose~~

→ Structured: it should be well structured. A well structured document is simple to understand and modify. A SRS document undergoes several revisions to core up with user requirement.

→ Black box view: —
it should define what the system should do and refrain from stating how to do these. this means that the SRS document should define the external behavior of the system and not discuss the implementation issues.

→ Conceptual integrity:- it should ~~do~~ show conceptual integrity so that the reader can merely understand it. Response to undesired events. it should characterized acceptable response to unwanted events. these are called system response to exceptional conditions.

→ Verifiable:- All requirement of the system as document in the SRS document should be correct. ~~this~~ this means that it should be possible to decide whether or not requirement have been met in an implementation.

Verification ≠ Validation

Verification ≠ validation is the process of investigation that a S/w system satisfies specifications and standards and it fullfills the required purpose. Barry Boehm described verification & validation as following.

Verification: Are we building product right?

Validation: Are we building the right product?

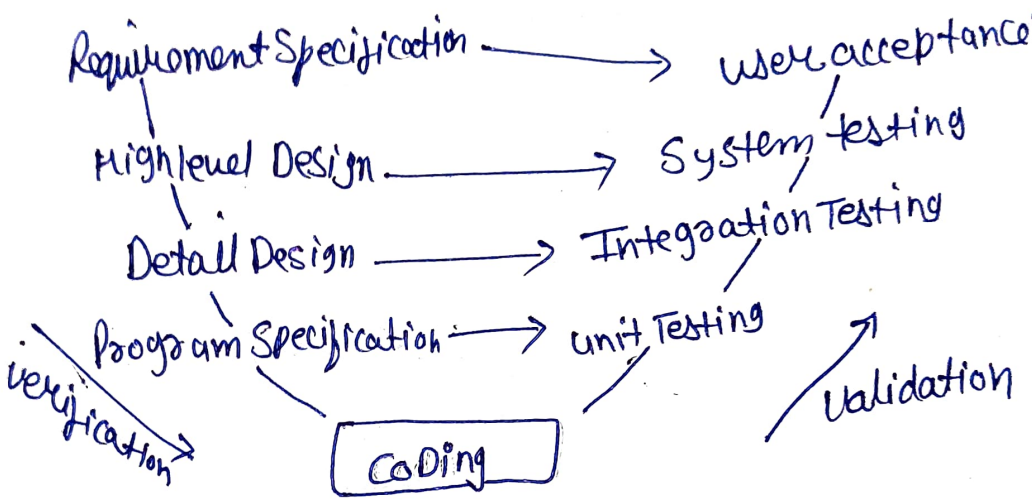
Verification:-

Verification is the process of checking that a S/w achieves its goals without any bugs. it is the process to ensure whether the product that is developed is right or not. it verifies whether the developed product fulfill the requirement that customer have.

Verification is Static Testing

Activites in verification invalved:

→ Inspection
→ Review
→ walkthrough
→ Desk Checking



Validation:

Validation is the process of Checking whether the S/w product is upto mark or in the ~~mark~~ other words product has high level requirements.

Validation is the dynamic testing

Activites invalved in validation process

Black box testing
White box testing
Unit testing
integration testing

verification followed by validation { verification → validation

Verification	V/S	Validation
→ it include checking document, design, codes & programs.		→ it includes ~~and~~ testing and validation the actual product.
→ verification is the ~~testing~~ static testing		→ validation is the dynamic ~~testing~~
→ it doesnot include the execution of the code.		→ it include the execution of the code
→ methods used in verification are review, walkthrough, inspection and desk Checking		→ methods used in validations are Black box ~~testing~~, whitebox ~~testing~~ and non functional testing.
→ quality assurance team does verification		→ validation is executed on s/w code with the help of testing team.
→ verification is about process, standard and guidline		→ validation is about the product