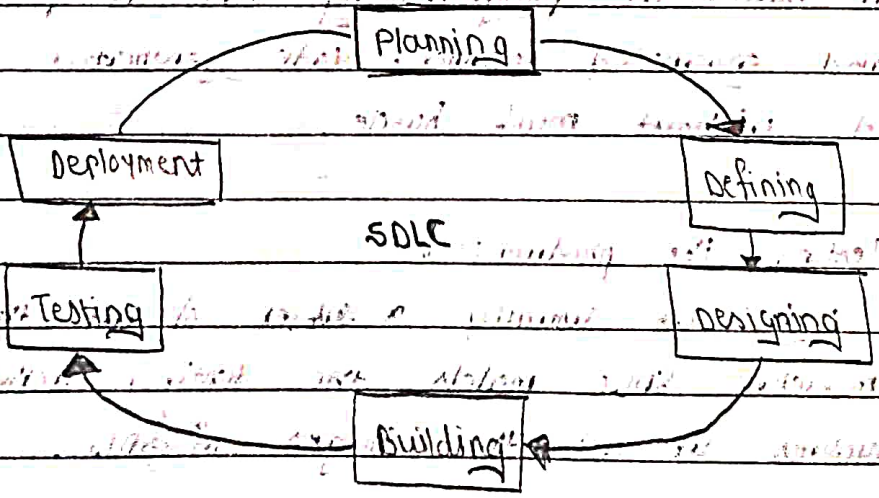
* Software Development life cycle :⟶

It is a process used by software industry to design, develop and test high quality softwares. The SDLC aims to produce a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of software development life cycle
- It is also called as software development process
- SDLC is a framework defining tasks performed at each steps in software development process
- ISO/IEC 12207 is an international standard for software life cycle process. It aims to be the standard for software that defines all the tasks required for developing and maintaining software.

SDLC is a process followed for a software project within the software organization. It consist of a detailed plan describing how to develop, maintain, replace and alter or enhance specific s/w. The life cycle defines a methodology for improving the quality of software and all the overall development process.

The following figures is a graphical representation of the various stages of a typical SDLC.



Stage 1: Planning and Requirement Analysis :—>

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with input from the customer, the sales department, market surveys and domain experts in the industry.

Stage 2: Defining Requirements :—>

Once the requirement analysis is done the next step is to clearly define and document in the product requirements & get them approved from the customer or the market analysis.

Stage 3: Designing the product Architecture :—>

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirement specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - design document Specification.

Stage 4: Building or developing the product :—>

In the stage of SDLC the actual development starts & product is built. The programming code is performed in a detailed and organised manner, code generation can be accomplished without much hussle.

Stage 5: Testing the product :—>

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involves in all the stages of SDLC.

Stage 6: Deployment in market & maintenance :—>

Once the product is tested and ready to be deployed it is released formally in the market sometimes product deployment happens in the stages as per the business strategy of that organization.

* Various software models :→

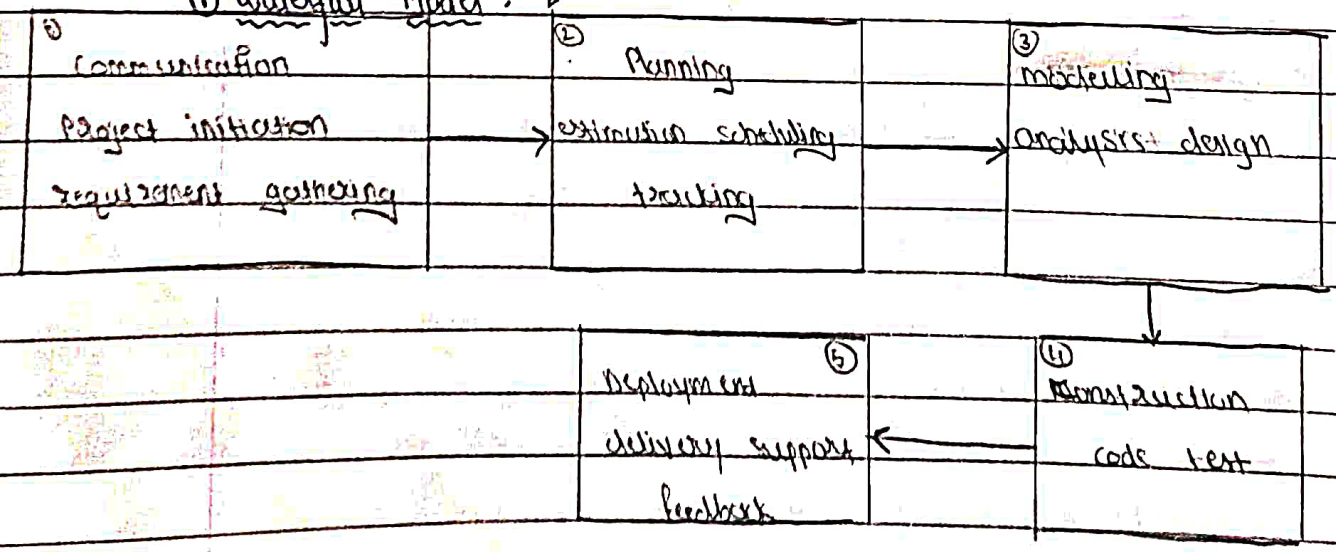
Software Development life cycle (SDLC) is a spiritual model used in project management that defines stages include in a information system development project from an initial feasibility study to the maintenance of the completed application.

There are different software development life cycle models specify and design, which are followed during the software development phase.

Here are some important phases of SDLC life cycle :

SDLC (models)
- Waterfall model
- RAD model
- Spiral Model
- V model
- Incremental model
- Agile model
- Iterative model
- Bigbang model

① Waterfall Model :→



- During the S/w development there is a situation when the requirements for problem are well understood when the work for communication to the deployment.
- waterfall model is called classic life-cycle model.
- It suggest systematic sequential approach. It starts from customer's specification for requirement gathering, modeling planning, construction, deployment.
- The waterfall model is oldest model.
- The real projects based upon waterfall model may follow sequential flow, it may sometimes cause diversions as the project team needs.
- The customer should also have patience a working version of program will not be available till it is trusted and made error free.
- The waterfall model is often inappropriate as it supports linear flow of software development. It served as useful process model where the situation requirements are fixed.

② RAD model →

- RAD model Rapid application development model.
- It is a linear sequential s/w development process model that emphasizes a cause development cycle using an element based construction approach.
- If the requirements are well known and understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a cause time period.
- RAD is a concept that products can be developed faster and of higher quality.

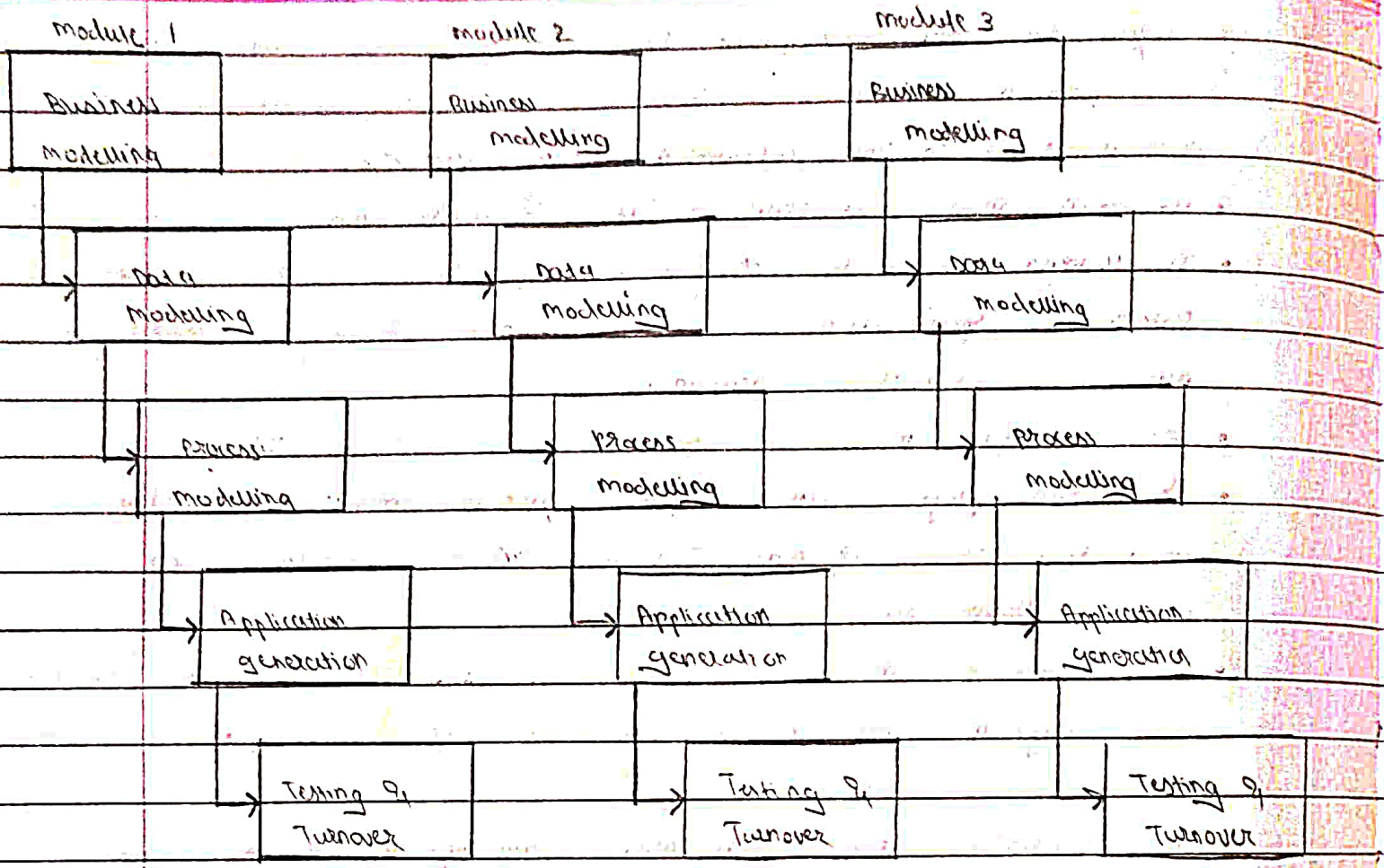


Fig: RAD model

- The various phases of RAD model are as follows:
  1. Business modelling
  2. Data modelling
  3. Process modelling
  4. Application generation
  5. Testing & Turnover
- Each phase in RAD brings highest priority functionality to the customer.
- For smaller projects, we cannot use the RAD model.
- If Application is not compatible with RAD and on the high technical risk, it's not suitable.

③ Spiral model :→

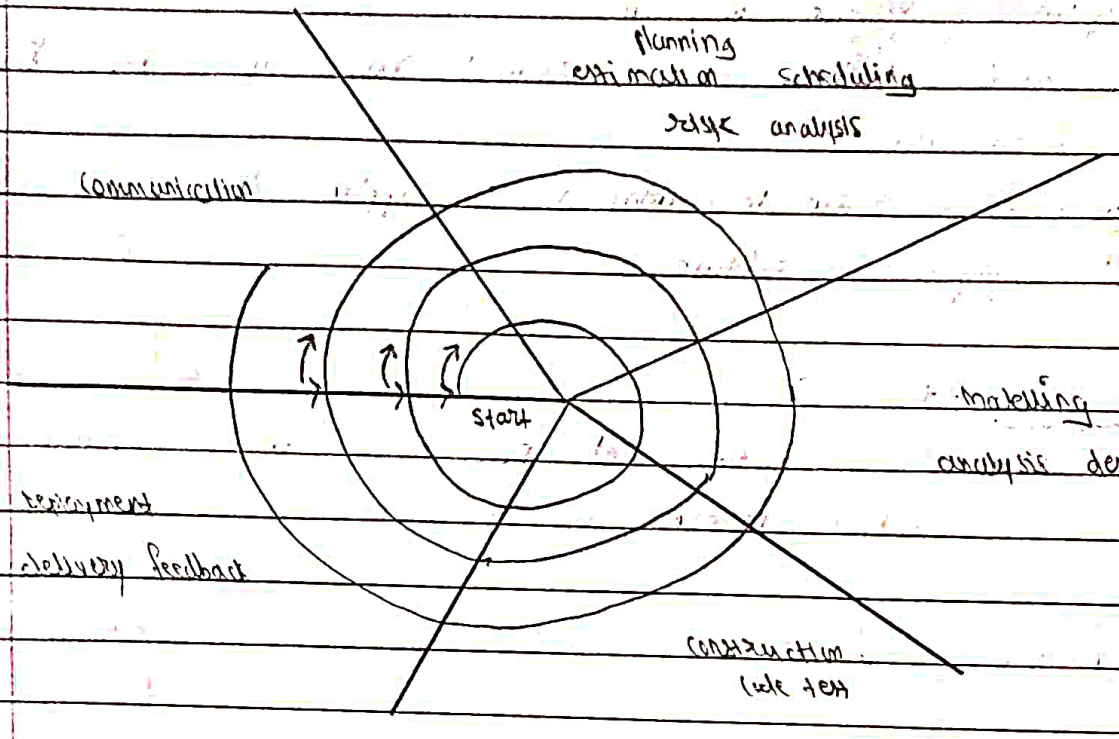


Fig : Spiral model

- Spiral model is an evolutionary s/w process model.
- It consist of iterative nature of prototyping along with a control and systematic aspect of waterfall model.
- It provides the potential for the rapid development of increasingly more complex version of the s/w.
- During early stage the release might be a model or prototype.
- During later stage increasingly more complex version of engineered system are produced.
- This model is divided into set of framework activities defined by s/w engineering team.
- As the evolutionary process begins the s/w team performs an activity implied by the circle. Spiral in the clockwise direction begining at the center.
- The first circuit around the spiral might preserved in development of product specification. Subsitutent passes around

the spiral might be used to develop a prototype, the more updated version of s/w.

- cost and schedule are adjusted based upon the feedback from the customer.
- Spiral model can be adapted to apply throughout the life of computer software.

④ V-model :→

- The V-model is a type of SDLC model where process executes in a sequential manner in V-shape.

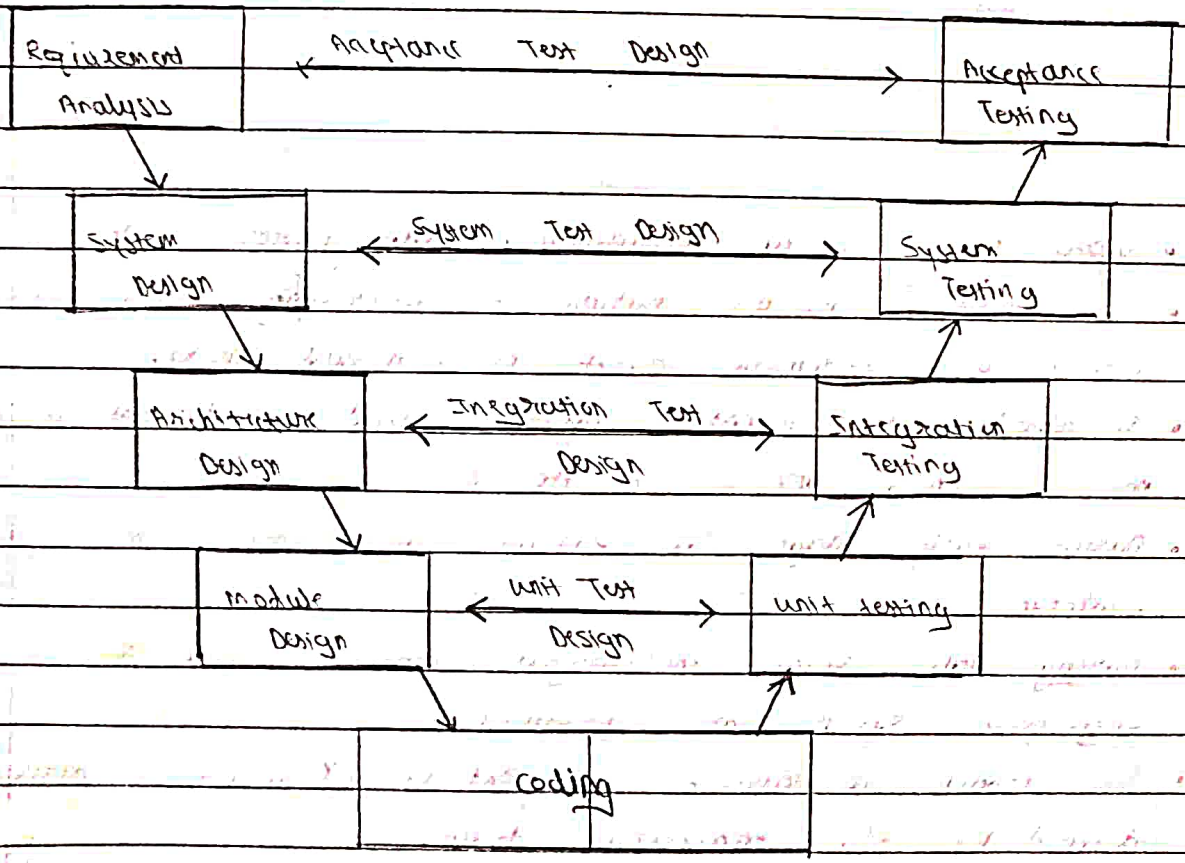


Fig : V-model

- It is also known as verification and validation model.
- Verification : It involves static analysis technique done without executing code. It is the process of evaluation of the product

development phase to find whether specified requirement meet.
- Validation : It involves dynamic analysis technique testing done by executing code. Validation is the process to evaluate the s/w after the completion of the development phase to determine whether s/w meet the customer expectation and requirements.
- The V-model is used when ample technical resources are available with technical expertise.
- It is not good for complex and object oriented project.

⑤ Incremental model :—▷
- The incremental model is not separated model.
- It is necessarily a series of waterfall cycles.
- The requirement are divided into group at start of the project for each group.
- SDLC model is followed to develop software.
- The SDLC process is repeated with each release adding more functionality until all requirement are met.

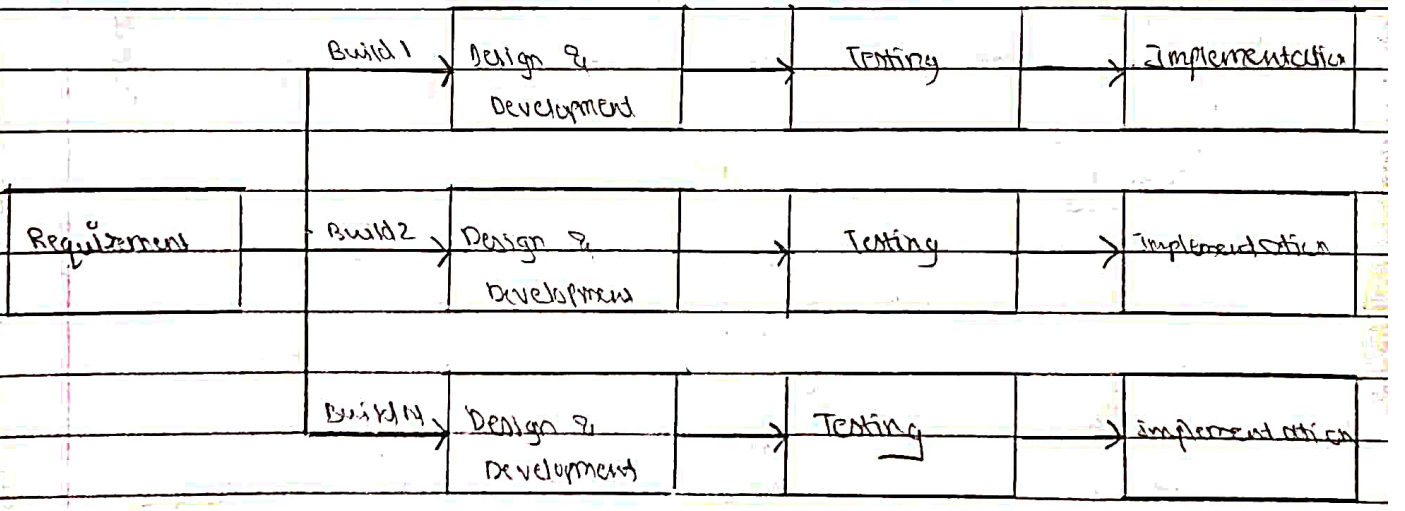
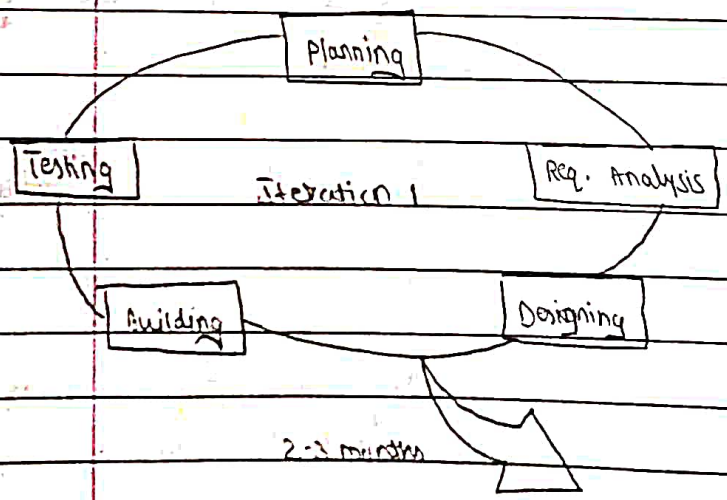
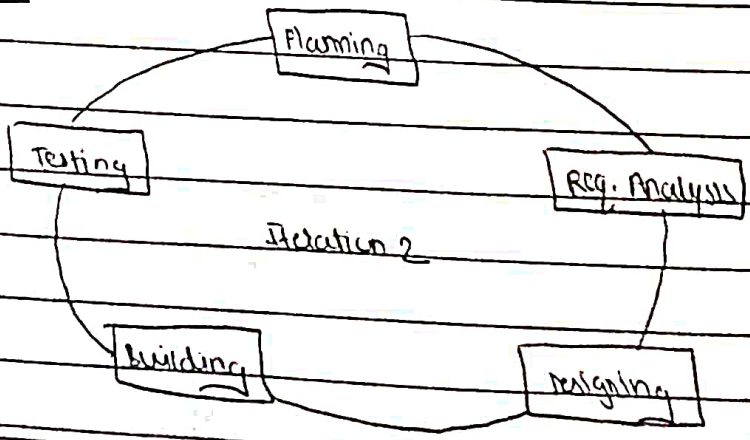


Fig: Incremental model

(6) Agile model :-

- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working sw product.
- Agile methods break the product into small incremental builds.
- In this, every iteration involves cross functional teams working simultaneously on various areas like -
  - * Planning
  - * Requirement Analysis
  - * Design
  - * Coding
  - * Unit Testing
  - * Acceptance Testing.
- Agile is base on the adaptive sw development methods.
- It is very realistic approach to sw development.
- It is easy to manage, gives flexibility to developers, no planning required.





③ Iterative model :→

- In this model, you can start with some of the s/w specification and develop the first version of s/w.
- Every release of iterative model finishes in an exact and fixed period that is called iteration.
- The final output of project renewed at the end of the s/w development life cycle process.
- Various phases in iterative model:

* Requirement gathering & analysis
* Design
* Implementation
* Testing
* Deployment
* Review
* Maintenance

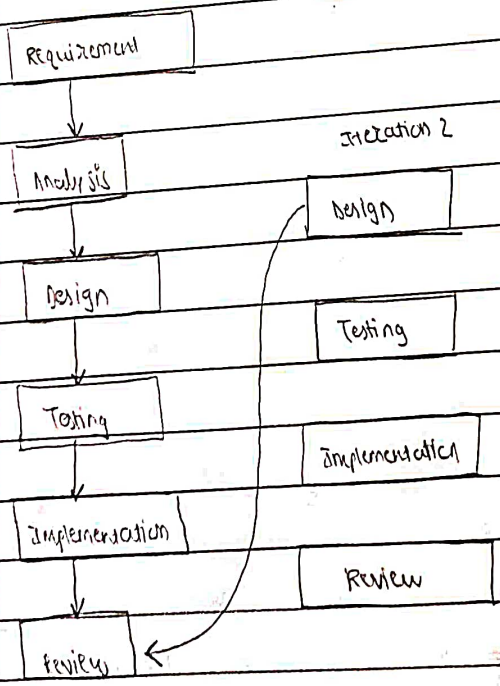


fig: Iterative model

Page No. 12

⑧ Bigbang model :→

- The bigbang model is an SDLC model where we do not follow any specific process.
- It does not follows a process / procedure and there is a very little planning required.
- Usually this model is followed for small project where the development teams are very small.
- It is an ideal model for the product where requirement are not well understood and the final release date is not given
- The Big Bang model is a very high risk model and changes in the requirement or misunderstood requirement may even leads to complete reversal or scrapping of project.

* System Engineering :—>

System Engineering is a transdisciplinary and integrative approach to enable the successful realization use and retirement of engineered systems using system principles and concepts and scientific technological and management methods.

We use the term "engineering" and "engineered" in their widest sense : "the action of working artfully to bring something about". "Engineered Systems" may be composed of any or all people, product, services, information, processes and natural elements.

System Engineering focuses on :

- Establishing, balancing and integrating stakeholder's goals, purpose and success criteria and defining actual or anticipated customer needs, operational concept and required functionality, starting in the development cycle.
- Establishing and appropriate lifecycle model, process approach and governance structures, considering the level of complexity, uncertainty, change and variety.
- Generating and evaluating alternative solution concepts and architectures.
- Baselining and modelling requirement and selected solution architecture for each phase of the endeavour.
- Performing design synthesis and system verification & validation.
- While considering both problem and solution domains taking into accounts necessary enabling systems and services, identifying the role that the parts play with respect to the overall behaviour and performance of the system and determining how to balance all of these factors to achieve a satisfactory outcomes.

System Engineering provides facilitation, guidance and leadership to integrate the relevant discipline and specialty groups into a cohesive effort, forming an appropriately structured development process that proceeds from concept to production operation, evolution and eventual disposal.

System Engineering considers both the business and the technical needs of customers with the goal of providing a quality solution that meets the needs of users and other stakeholders, is fit for the intended purpose in real-world operation and avoids or minimized adverse unintended consequences.

* Abstraction ...

Date
Page 15

* Requirement Engineering :→

Designing and building computer software is challenging. In fact, building software is so compelling that many software developers want to jump right in before they have a clear understanding of what is needed. The broad spectrum of tasks and techniques that lead to an understanding of requirements is called requirement engineering. From a software process prespective requirements engineering is a major software engineering action that begins during the communication activity and continues into modeling activity.

Requirement engineering builds a bridge to design and construction. But when does the bridge originate. One could argue that it begins at the feet of the project the stakeholder, where business need is defined, user scenarios are described functions and features are delineated.

Inception :→ How does a s/w project get started? Is there a single event that becomes the catalyst for a new computer-based system or product or does the need evolve over time? There are no definitive answers to these questions.

Elicitation :→ It certainly seems simply enough- ask customer, the users and other what the objectives for the system or product are, what is to be accomplished, how the system or product fits into the needs of the business and finally how the system or product is to be used on a day-to-day basis.

Problem of scope :→ The boundary of the system is indefined or the customers / user specify unnecessary technical detail that may

confuse rather than clarify, overall system objectives.

problem of understanding :→ The customers / users are not completely sure of what is needed have a poor understanding of capabilities and limitations of their computing environment, don't have a full understanding of the problem domain have trouble communicating needs to the system engineering.

Problem of volatility :→ The requirements change over time.

Specification :→ In the context of computer based system the term specification means different things to different people. A specification can be a written document a set of graphical models, a formal mathematical model or any combinating these.

* Abstraction Architecture :—>

• Abstraction :

When you consider a modular soln to any problem many levels of abstraction can be posed. At the highest level of abstraction a solution is stated in broad terms using the language of the problem environment.

A data abstraction is a named collection of data that describes a data object. In the context of procedural abstraction open, we can define a data abstraction called door. Like any data object the data abstraction for door would encompass a set of of attributes that describe the door.

• Architecture :

Software architecture means the overall structure of the s/w and the ways in which that structure provides conceptual integrity for a system. In its simplest form architecture is the structure or organization of program components. The manner in which these component interact and the structure of data that are used by the components.

One goal of s/w design is to derive an architectural rendering of a system. This rendering serves as a framework from which is more detailed design activities are conduced.

A set of architectural pattern enables a s/w engineer to solve common design problems.

• structural properties :

This aspect of the architecture design representation defines the components of a system and the manner in which those components are packaged and interact with one another.

The architectural design description should drawn upon repeatable patterns that are commonly encounters in the design of families of similar system in essence, the design should have the ability to reuse architectural building blocks

* Pattern modularity :—

• Pattern :—

A pattern is a named insight which conveys the essence of a proven solution to a recurring problem within a certain context and system of forces. A design pattern describe a design structure that solve a particular design problem within a specific context and may have an impact on the manner in which the pattern is applied and used.

The intent of each design pattern is to provide a description that enables a designer to determine whether the pattern is applicable to the current work whether the pattern can be reused.

• modularity :—

Modularity is the most common manifestation of separation of concerns. S/w is divided into separately named and addressable component, sometimes called modules, that are integrated to satisfy problem requirement.

It has been stated that modularity is the single attribute of s/w that allows a program to be intellectually manageable. monolithic s/w cannot be easily grasped by a s/w engineer. The number of control paths, span of reference number of variables and over all complexity would make understanding close to impossible.

You modularize a design so that development can be more easily planned; software increments can be defined and delivered; changes can be easily accomodated; testing and debugging can be conducted more efficiently, and long term maintenance can be conducted without serious effects.

* White - Box Testing :→

White Box Testing is a testing technique in which software's internal structure, design and coding are tested to verify input, output flow and improve design, ability and security. In white box testing, code is visible to testers so it is also called as clear box testing, open box testing, transparent box testing, code based testing and glass box testing.

In one of two part of Box testing approach to software testing. Its counter part, Black box testing, involves testing from an external or end-user perspective. On the other hand, white box testing is also engineering based on the inner working of an application and involves around internal testing.

The term "white Box" was used because of the see-through box concept. The clear box or white box name symbolizes the ability to see through the software outer shell (or "box") into its inner workings.

Like this, white Box testing involves the testing of the software code for the following:

- Internal security holes.
- Broken or poorly structured paths in the coding processes.
- The flow of specific input through the code
- Expected outputs.
- The functionality of conditional loops.
- Testing of each statement, object, and function on an individual basis.

* Black-Box Testing :—>

Black Box testing also called behaviour testing focuses on the functional requirements of the software. That is, black box testing technique enables you to derive sets of input condition that will fully exercise all functional requirement for a program. Black box testing is not an alternative to white Box testing. Rather, it is a complementary approach that is likely to uncover a different class of error than white box methods.

Black box testing attempts to find error in the following categories :

- incorrect or missing function
- interface errors
- errors in data structure or external database access
- behaviour or performance error
- initialization and termination error.

Black box testing tends to be applied during later stage of testing. Because black box testing purposely disregards control structure, attention is focused on the information domain. Tests are designed to answer the following questions

- How is functional validity tested?
- How are system behaviour and performance tested?
- How are the boundaries of a data class isolated?
- what cases of input will make good test cases?
- what data rates and data volume can be the system tolerate?

By applying black box techniques, you derive a set of test cases that satisfy the following criteria :

- Test cases that reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieved ~~reasonable~~ reasonable testing.
- Test cases that tell you something about the presence or absence of classes of errors rather than an error associated only with the specific test at hand.

* Metrics for software quality :—

The overriding goal of sw engineering is to produce a high-quality system, application or product within a time frame that satisfies a market need. To achieve this goal, sw engg must apply effective methods coupled with a mature sw process.

A project manager must also evaluate quality as the project progresses. Private metrics collected by individual sw engineers are combined to provide project-level results. Although many quality measures can be used the primary thrust at the project level is to measure errors and defects. Metrics derived from these measure provide an indication of the effectiveness of individual and group sw quality assurance and control activities.

Software metrics can be classified into two types as follows:

- Product metrics: These are the measures of various characteristic of the sw product. The two important sw characteristics are 1) Size and complexity of software.
  2) Quality and reliability of software
- Process metrics: These are the measures of various characteristic of sw development process.

Measuring Quality:

- Correctness: A program must operate correctly or it provide little value to its user. Correctness is the degree to which the sw performs its required function.
- Maintainability: Sw maintenance and support account for more effort than any other sw engineering activity.

- Integrity : S/w integrity has become increasingly important in the age of cyber terrorists and hackers. This attributes measure a system ability to with-stand attack to its security.

- Usability : If a program is not easy to use, it is often classed to failure, even if the function they it performs are available vulnerable usability is an attempt to quantify ease of use and can be measured intern.

* Project Management :→

Software in project management is dedicated to the planning, scheduling, resource allocation, execution, tracking and delivery of software and web project.

Project management in sw engineering is distinct from traditional project management - Software in project management has a unique life cycle process that requires multiple rounds of testing, updating and customer feedback.

Software project managers may have to do any of following tasks:

- planning :→ The sw project manager puts together the blueprint for the entire project. The plan will define the scope, timeline, communication strategy, testing and maintenance.
- Leading :→ A sw project manager assembles and leads the project team, which consist of developers, analysts, testers, graphic designers and technical writers.
- Execution :→ The person who manages software projects will supervise the successful execution of each stages of the project. This includes monitoring progress, conducting frequent team checkins and creating status reports.
- Time management :→ Staying on schedule is crucial to the successful completion of any project. Sw project manager must be expert in risk management and contingency planning to ensure progress in the face of roadblocks or changes.
- Budget :→ Like traditional project managers, professionals who manage sw projects are tasked with creating a budgets for a project and sticking to it as closely

as possible, moderating spending and reallocating funds when necessary.

- Maintenance :⟶ Project management in SDLC encourages constant product testing to discover and fix bugs early, adjust the end product to the customer's needs, and keep the project on target. The SDLC project manager ensures the product is properly and consistently tested, evaluated and adjusted accordingly.

How to manage a SDLC project successfully :⟶

- Taking non-development work off your team's plate to let them focus on the product.
- Motivating your team by sharing other's success stories.
- Avoiding any changes to tasks once assigned.
- Encouraging organization by being organized yourself.
- Breaking down the plan and assigned specific daily task.

* Reengineering - Software reengineering :→

Software reengineering is a process of software development which is done to improve the maintainability of software system. Reengineering is the examination and alteration of the system to reconstitute it in a new form.

* Objectives of Re-engineering :-

- To derive a cost effective option for system evolution.
- To describe the activities involved in the S/w maintenance process.
- To distinguish betⁿ S/w and data reengineering to explain the problems of data reengineering.

* Steps involved in Reengineering :-

① Inventory Analysis
② Document Reconstruction
③ Reverse Engineering
④ Code Reconstruction
⑤ Data Reconstruction
⑥ Forward Engineering

* Reengineering cost factor :-

- The quality of the software to be re-engineered
- The total support available for reengineering.
- The extend of the required data conversion
- The availability of expert staff for re-engineering.

* Advantages of Reengineering :-

① Reduce Risk : As the S/w is already existing, the risk is less as compared to new S/w development.

(2) Reduced cost : The cost of reengineering is less than the cost of developing a new software.

(3) Revelation of Business Rules : As a system is reengineered business rules that are embedded in the system are rediscovered.

(4) Better use of existing staff : Existing staff expertise can be maintained and extended accomodate new skills during reengineering.

* Disadvantage of Re-engineering :-

- Practical limits to the extent of reengineering.
- Major architectural changes or radical reorganizing of the system data management has to be done manually.
- Reengineered system is not likely to be as maintainables as a new system developed using modern sw reengineering methods.

* Reverse Engineering :→

Software reverse engineering is a process of recovering the design, requirement specification and functions of a product from an analysis of its code. It builds a program database and generates information from this.

The purpose of reverse engineering is to facilitate the maintenance work by improving the understandability of a system and to produce the necessary document for a legacy system.

* Reverse Engineering Goals :

- Cope with complexity
- Recover lost information
- Detect side effects
- Synthesize higher abstraction
- Facilitate Reuse.

* Steps of Software Reverse Engineering :

① Collection Information :-

This step focuses on collecting all possible information about the software.

② Examining the information :-

The information collected in step 1 as studied as to get familiar with the system.

③ Extracting the Structure :-

This step concern with identification of program structure chart where each node correspond to some routins.

④ Recording the functionality :→

During this step processing details of each module of the structure charts are recorded using structured language like decision table and etc..

⑤ Recording control flow :→

High level control structure of software is recorded.

* Reverse Engineering Tools:

Some of the tools are given below:

• GRAS and GIA :→

A graphical navigator for software and web repositories along with a collection of reverse engineering tools.

• Rigi :→

A visual software understanding tool.

• Bunch :→

A s/w clustering / modularization tool.

• GEN++ :→

An application generator to support development of analysis tools for the C++ language.

• PBS :→

software Bookshelf tools for architecture of program.