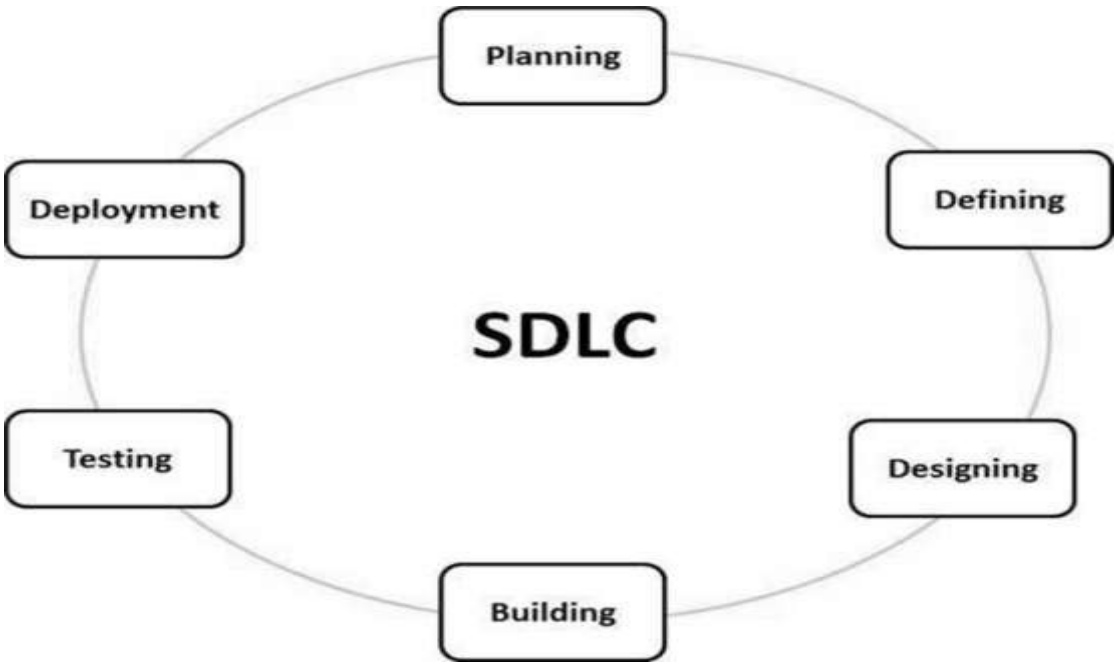
PRACTICAL 2

AIM: Identify Suitable Design and Implementation model from the different software engineering models.

SOFTWARE PROCESS MODELS:

- The process model is the abstract representation of process.
- Also known as Software development life cycle (SDLC) or Application development life cycle Models
- Process models prescribe a distinct set of activities, actions, tasks and milestones (deliverables) required to engineer high quality software.
- Process models are not perfect, but provide roadmap for software engineering work.
- Software models provide stability, control and organization to a process that if not managed can easily get out of control.
- Software process models are adapted (adjusted) to meet the needs of software engineers and managers for a specific project.

SDLC PHASES:



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 inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

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 or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

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 requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all he stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

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 the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

LIST OF MODELS:

- Linear Sequential Waterfall model
- Prototype Model
- RAD Model.
- Evolutionary Software Process Model

    - Incremental Model
    - Spiral Model
    - Concurrent Development Process Model
    - Component based Assembly Model
- The Formal Methods Model
- Fourth Generation Techniques

DIFFERENCE BETWEEN MODELS:

Properties of Model	Water-Fall Model	Incremental Model	Spiral Model	Rad Model
Planning in early stage	Yes	Yes	Yes	No
Returning to an earlier phase	No	Yes	Yes	Yes
Handle Large-Project	Not Appropriate	Not Appropriate	Appropriate	Not Appropriate
Detailed Documentation	Necessary	Yes but not much	Yes	Limited
Cost	Low	Low	Expensive	Low
Requirement Specifications	Beginning	Beginning	Beginning	Time boxed release
Flexibility to change	Difficult	Easy	Easy	Easy
User Involvement	Only at beginning	Intermediate	High	Only at the beginning
Maintenance	Least	Promotes Maintainability	Typical	Easily Maintained
Duration	Long	Very long	Long	Short
Risk Involvement	High	Low	Medium to high risk	Low
Framework Type	Linear	Linear + Iterative	Linear + Iterative	Linear

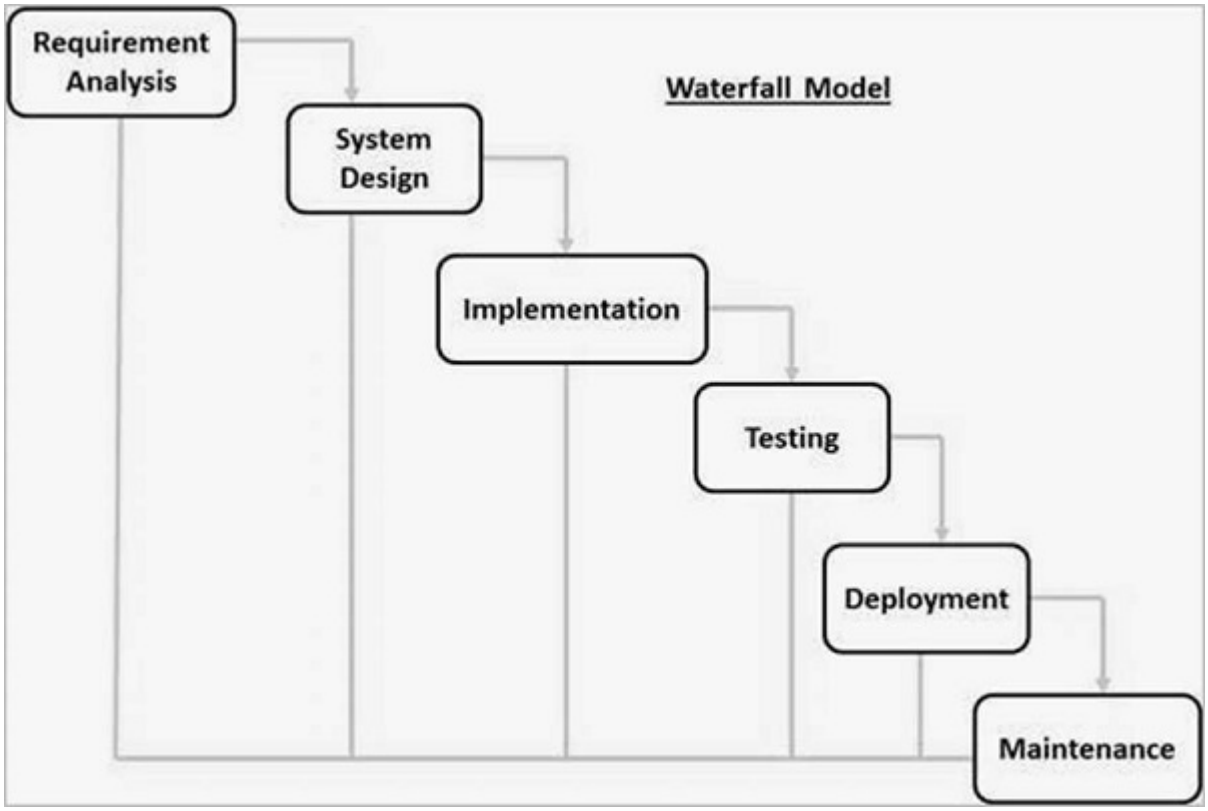
Testing	After completion of coding phase	After every iteration	At the end of the engineering phase	After completion of coding
Overlapping Phases	No	Yes (As parallel development is there)	No	Yes
Maintenance	Least Maintainable	Maintainable	Yes	Easily Maintainable
Re-usability	Least possible	To some extent	To some extent	Yes
Time-Frame	Very Long	Long	Long	Short
Working software availability	At the end of the life-cycle	At the end of every iteration	At the end of every iteration	At the end of the life cycle
Objective	High Assurance	Rapid Development	High Assurance	Rapid development
Team size	Large Team	Not Large Team	Large Team	Small Team
Customer control over administrator	Very Low	Yes	Yes	Yes

DIAGRAM FOR SELECTED MODEL AND ITDESCRIPTION:

- ## Waterfall Model
  - The waterfall model is the classical model of software engineering.
  - This model emphasizes planning in early stages, it ensures design flaws before they develop.
  - In addition, its intensive document and planning make it work well for projects in which quality control is a major concern.
  - When requirements for a problems are well understood then this model is used in which work flow from communication to deployment is linear
  - **When to use ?**
    - Requirements are very well known, clear and fixed
    - Product definition is stable
    - Technology is understood
    - There are no ambiguous (unclear) requirements

- Ample (sufficient) resources with required expertise are available freely
- The project is short.

DIAGRAM OF 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.

ADVANTAGES:

- Simple & Easy
- Easy to manage
- Works well for smaller projects
- Results are well documented

DRAWBACKS:

- Risk & Uncertainty
- Not for projects where requirements are changing
- Working version is not available during development. Which can lead the development with major mistakes.
- Deadlock can occur due to delay in any step.
- Not suitable for large projects.
- Not for big & complex projects

JUSTIFICATION: THAT WHY YOU CHOOSE THIS MODEL ONLY:

The Reason we choose waterfall model was:

- We are developing the project for small scale, and requirements are fixed i.e.no ambiguous (unclear) requirements.
- Results are well documented.

- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

DIAGRAM OF MODEL ACCORDING TO YOUR DEFINATION: