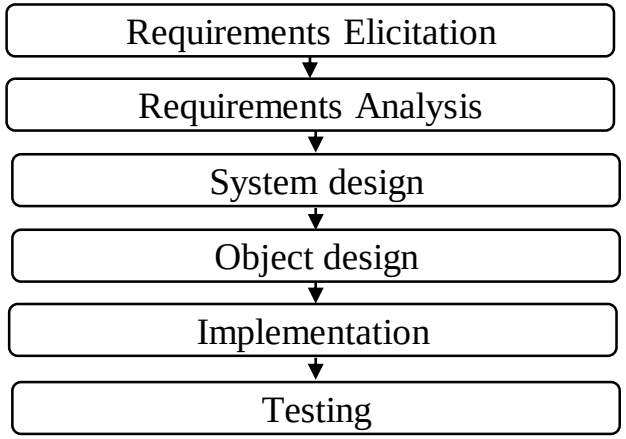
Object-Oriented Analysis and Design using JAVA (20B12CS334)

B.Tech (CSE/IT) 5th SEM
2021-2022

Lecture-6: Object-oriented software development life cycle (SDLC)

Introduction

The *IEEE Standard for Software Life Cycle Processes* describes the set of activities and processes that are mandatory for the development and maintenance of software.



An **activity** is a task or group of sub activities that are assigned to a team or a project participant to achieve a specific purpose.	A **process** is a set of activities that is performed toward a specific purpose.
The *Requirements Process*, for example, is composed of three activities: • *Define and Develop Software Requirements* during which the functionality of the system is defined precisely • *Define Interface Requirements* during which the interactions between the system and the user are defined precisely • *Prioritize and Integrate Software Requirements* during which all requirements are integrated for consistency and prioritized by client preference.	Examples of processes in the development process group include • the *Requirements Process* during which the developers develop the system models • the *Design Process* during which developers decompose the system into components • the *Implementation Process* during which developers realize each component.

Referred sources mentioned in Key References

Requirement elicitation

1. During **requirements elicitation**, the client and developers define the purpose of the system.

1. The result of this activity is a description of the system in terms of actors and use cases.

1. Actors represent the external entities that interact with the system. Actors include roles such as end users, other computers the system needs to deal with (e.g., a central bank computer, a network), and the environment (e.g., a chemical process).

1. Use cases are general sequences of events that describe all the possible actions between an actor and the system for a given piece of functionality.

Requirement analysis

1. During **analysis**, developers aim to produce a model of the system that is correct, complete, consistent, and unambiguous.

1. Developers transform the use cases produced during requirements elicitation into an object model that completely describes the system.

1. During this activity, developers discover ambiguities and inconsistencies in the use case model that they resolve with the client.

1. The result of analysis is a system model annotated with attributes, operations, and associations.

1. The system model can be described in terms of its structure and its dynamic interoperation

System design

1. During **system design**, developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams.

1. Developers also select strategies for building the system, such as the hardware/software platform on which the system will run, the persistent data management strategy, the global control flow, the access control policy, and the handling of boundary conditions.

1. The result of system design is a clear description of each of these strategies, a subsystem decomposition, and a deployment diagram representing the hardware/software mapping of the system.

1. Whereas both analysis and system design produce models of the system under construction, only analysis deals with entities that the client can understand.

1. System design deals with a much more refined model that includes many entities that are beyond the comprehension (and interest) of the client.

Referred sources mentioned in Key References

Object design

1. During **object design**, developers define solution domain objects to bridge the gap between the analysis model and the hardware/software platform defined during system design.

1. This includes precisely describing object and subsystem interfaces, selecting off-the-shelf components, restructuring the object model to attain design goals such as extensibility or understandability, and optimizing the object model for performance.

1. The result of the object design activity is a detailed object model annotated with constraints and precise descriptions for each element.

Implementation

1. During **implementation**, developers translate the solution domain model into source code.

1. This includes implementing the attributes and methods of each object and integrating all the objects such that they function as a single system.

1. The implementation activity spans the gap between the detailed object design model and a complete set of source code files that can be compiled.

Referred sources mentioned in Key References

Testing

1. During **testing**, developers find differences between the system and its models by executing the system (or parts of it) with sample input data sets.

1. During unit testing, developers compare the object design model with each object and subsystem. During integration testing, combinations of subsystems are integrated together and compared with the system design model.

1. During system testing, typical and exception cases are run through the system and compared with the requirements model.

1. The goal of testing is to discover as many faults as possible such that they can be repaired before the delivery of the system.

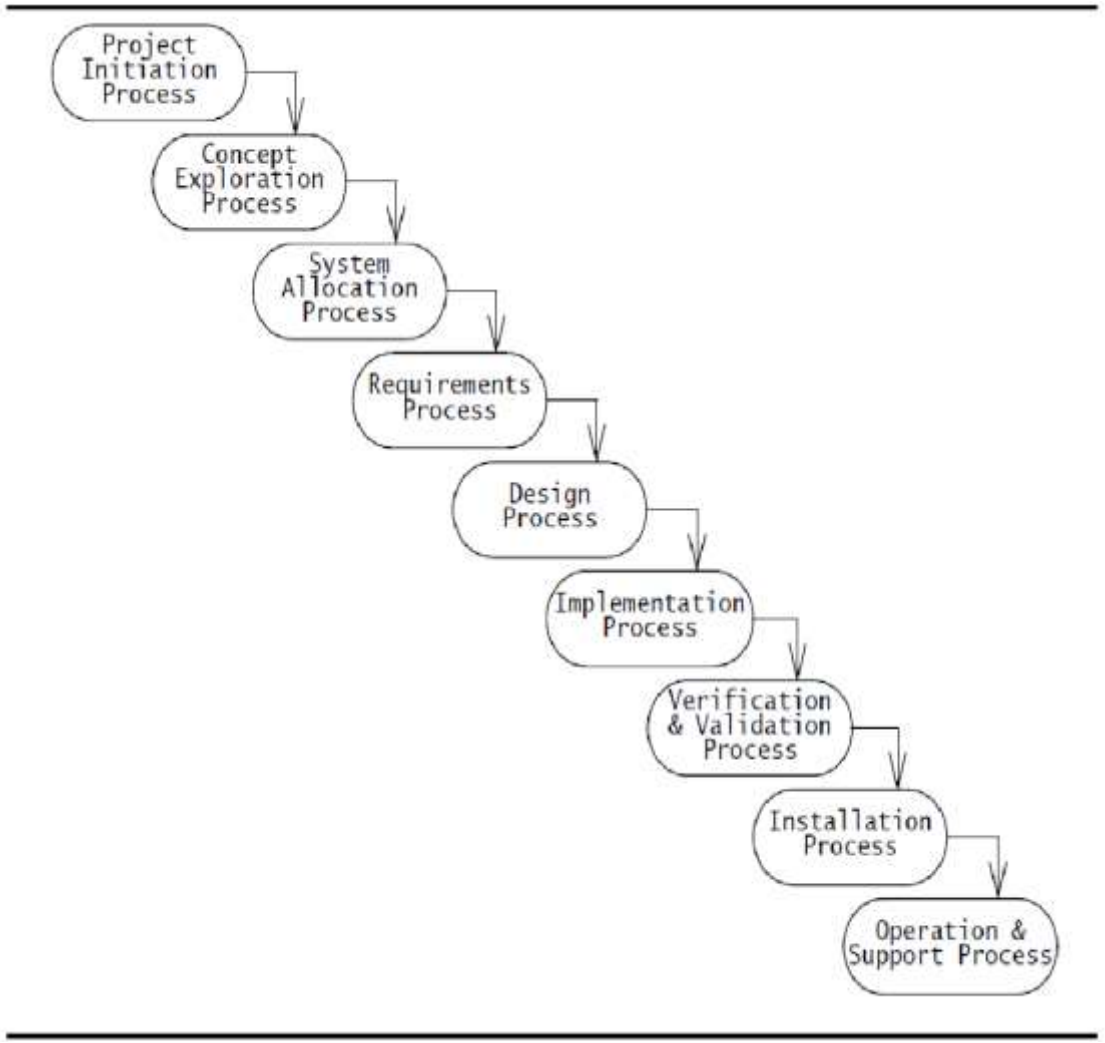
1. The planning of test phases occurs in parallel to the other development activities: System tests are planned during requirements elicitation and analysis, integration tests are planned during system design, and unit tests are planned during object design.

Waterfall model

The **waterfall model**, is an activity-centered life cycle model that prescribes sequential executions of subsets of the development processes and management processes.

All requirements activities are completed before the system design activity starts. The goal is never to turn back once an activity is completed. The key feature of this model is the constant verification activity which ensures that each development activity does not introduce unwanted or delete mandatory requirements.

This model provides a simple (or even simplistic) view of software development that measures progress by the number of tasks that have been completed. The model assumes that software development can be scheduled as a step-by-step process that transforms user needs into code.



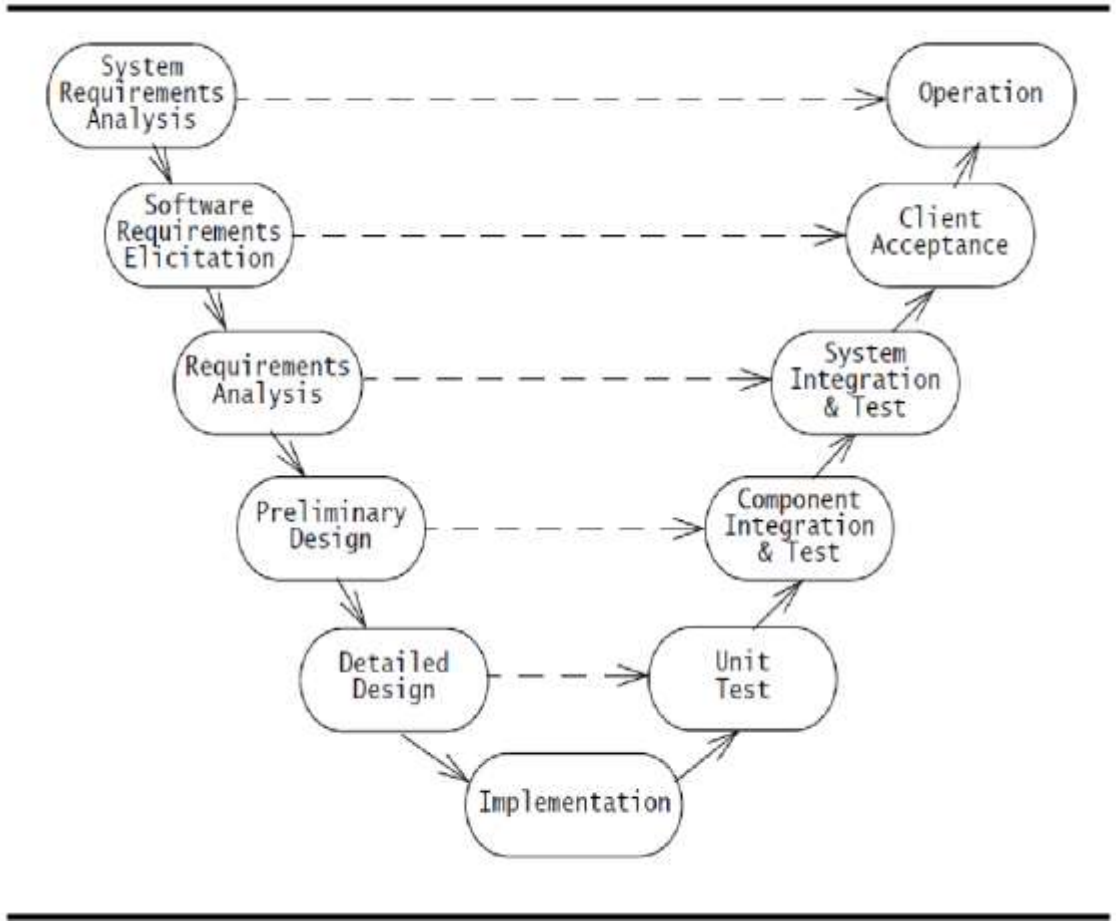
Referred sources mentioned in Key References

V-model

The **V-model** is a variation of the waterfall model that makes explicit the dependency between development activities and verification activities.

The difference between the waterfall model and the V-model is that the latter depicts the level of abstraction.

All activities from requirements to implementation focus on building an increasingly detailed representation of the system, whereas all activities from implementation to operation focus on validating the system.



Referred sources mentioned in Key References

Spiral model

The spiral model focuses on addressing risks incrementally, in order of priority. Each round is composed of four.

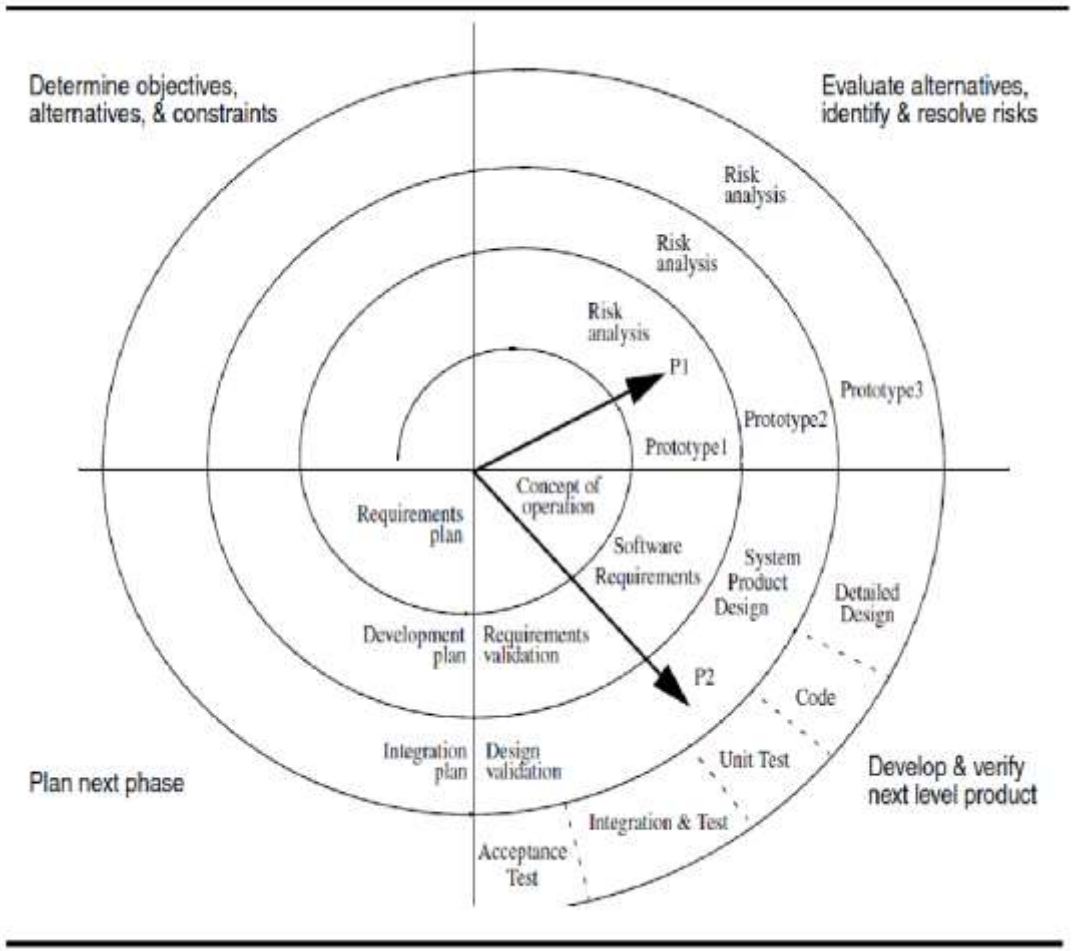
During the first phase (upper left quadrant), developers explore alternatives, define constraints, and identify objectives.

During the second phase (upper right quadrant), developers manage risks associated with the solutions defined during the first phase.

During the third phase (lower right quadrant), developers realize and validate a prototype or the part of the system associated with the risks addressed in this round.

The fourth phase (lower left quadrant) focuses on planning the next round based on the results of the current round.

The last phase of the round is usually conducted as a review involving the project participants, including developers, clients, and users. This review covers the products developed during the previous and current rounds and the plans for the next round.



Referred sources mentioned in Key References

Each round follows the waterfall model and includes the following activities:

1. Determine objectives
2. Specify constraints
3. Generate alternatives
4. Identify risks
5. Resolve risks
6. Develop and verify next-level product
7. Plan.

The first two activities define the problem addressed by the current cycle. The third activity, *Generate alternatives*, defines the solution space. The activities *Identify risks* and *Resolve risks* identify future problems that may result in high cost or cancellation of the project.

The activity *Develop and verify next-level product* is the realization of the cycle. The activity *Plan* is a management activity to prepare for the next cycle. The first round, *Concept of Operation*, starts in the upper left quadrant. Subsequent rounds are represented as additional layers on the spiral. The notation makes it easy to determine the status of the project at any time. The distance from the origin is the cost accumulated by the project. The angular coordinate indicates the progress accomplished within each phase

Key references

1. Object-Oriented Analysis and Design with Applications-Third Edition-Addition Wesley Authors-Grady Booch Robert A. Maksimchuk Michael W. Engle Bobbi J. Young, Ph.D. Jim Conallen Kelli A. Houston

Thank You