Introduction

Object Orientation

- **Object Orientation (OO)** is a **paradigm** that views and models **a system** as **a collection of interacting objects**.
- **Abstraction** is a model to **include the most important aspects** while **ignoring less important details**.
- **Encapsulation** is a mechanism for **restricting access** to some **internal components** of an **object**.
- **Polymorphism** is the ability of **different objects to respond to the same request in different ways**.

Objects

- An **object** corresponds to **a single entity** in the **real world**.
- **Objects** may be **tangible or intangible**.
- All **objects** contain **information** and **exhibit behavior**.

Classes

- A **class** is a **uniquely identified abstraction** of a set of **related instances** that share **identical or similar characteristics**.
- An **attribute** is a **named property** of a **class**.
- An **operation** is the **implementation** of a **service**.
- A **class** is an **object-blueprint** and becomes an **object** when **instantiated**.

Object Oriented Software Development

- A **software process** is a **set of activities** that lead to the **production** of **software**.
- A **software process model or paradigm** is an **abstraction** of **software processes**.
- Common software paradigms include:
  1. **Waterfall -** A **linear process** with **distinct phases**.
  2. **Spiral - Iterative risk management**.
  3. **Agile -** An **iterative** and **incremental** methodology.

OOA, OOD, OOP, and OOT

- **Object-Oriented Analysis (OOA)** is the process of **analyzing a problem** and **discovering all entities** associated with the problem.
- **Object-Oriented Design (OOD)** is the process of **taking the entities discovered in OOA** and **determining how they interact**.
- **Object-Oriented Programming (OOP)** is the process of **implementing an object-oriented design** in a **programming language**.
- **Object-Oriented Testing (OOT)** is the process of **testing the implemented design**.

The Unified Process

- The **Unified Process** is an **iterative and incremental software development process framework**.
- The **main principals** of the **Unified Process**:
  1. It is **use-case driven**. A **use case** is a **written description** of **interactions** between **a role** and **a system** to **achieve a goal**. It links the **requirements** to the **implementation**.
  2. It is **architecture-centric**. It is a **theme** from the **earliest stages** of a **project**.
  3. It relies on **workflow in iterations**.
  4. It creates **incremental development**. Each **iteration** has **four properties**: the **duration**, the **tasks**, the **outcomes**, and the **usage**.
- The **main benefit** to **iterative development** is the ability to get **feedback, on a regular basis**.

Project Inception

Project Inception

- The **first stage** in a project is the **inception stage**.
- The **inception stage** is a short stage that addresses the following questions:
  1. What are the outcomes?
     a. What is the vision and business case for the project?
     b. Is it feasible to work on this project?
     c. Should the project be purchased or built?
     d. What is the rough cost range of developing the project?
     e. Should the project continue, or stop?
  2. What are the methods for achieving the outcome?
     a. What are the requirements and how will they be achieved?
  3. What are the objectives?
     a. What are the most important objectives?
     b. What is the initial plan?

The Artifacts of Inception

- There are **9 artifacts of inception**:
  1. **Vision and Business Case -** Describes the high-level goals and constraints, the business case, and provides an executive summary.
  2. **Use Case Model -** Describes the functional requirements. During inception, the names of most use cases will be identified, and some will be analyzed in detail.
  3. **Supplementary Specification -** Describes other requirements, mostly non-functional requirements. During inception, it is useful to have an idea of the key non-functional requirements that have a major impact on the architecture.
  4. **Glossary -** The key domain terminology, and a data dictionary.
  5. **Risk List and Risk Management Plan -** Describes the risks (business, technical, resource, schedule) and ideas for their mitigation.
  6. **Prototypes and proof-of-concepts -** Clarifies the vision, and validates the technical ideas.

7. **Iteration Plan** - Describes what to do in the first elaboration iteration.
8. **Phase Plan and Software Development Plan** - Low-percision guess for the elaboration phase duration (tools, people, education, resources).
9. **Development Case** - A description of the customized UP steps and artifacts for the project.

The Vision Document

- A **vision document** is a document that **describes an idea or project**. It defines the product / service to be developed in terms of the **stakeholder's key needs**.
- There are **6 sections** in the **vision document**:
  1. **Introduction** - Describe the project with one or two lines.
  2. **Problem Statement** - Use a short paragraph to explain the problem that is being solved.
  3. **Stakeholders** - Identify stakeholders (owner, manager, customer, etc) and their key interests (what they need to be able to do).
  4. **User and Goals** - Identify the users and user-level goals (users are usually stakeholders).
  5. **Summary** - List the system's functional (services) and non-functional (constraints) requirements.
  6. **Project Risk** - Explain what might be difficult to design, and why.

The List Of Requirements and Glossary

- The **list of requirements** states the **main requirements** that solution must contain, and assigns them each a unique number (R1, R2, ...).
- The **glossary (data dictionary)** defines all **terms that will be used throughout the project** as well as any **alias** they may have.