

UNIT-7

Design and Implementation

⊗ Introduction: Software design is the stage at which the conceptual/logical model is converted to physical model. It is the process by which an agent creates a specification of a software outlook intended to accomplish goals, using a set of components.

Implementation is the process of realizing the design as a program. It is a systematically structured approach to transform the design model into a working product.

⊗ Object Oriented Design using the UML:

Object oriented design is a means of designing software so that the functional components in the design represent objects with their own private states and operations rather than functions. This process involves designing the object classes and the relationship between these classes. Software that are developed using object oriented design are easier to change than the systems developed using functional approaches. There are a variety of different object-oriented design processes that depend on the organization using the process. Common activities in these processes include:

- Define the context and modes of use of the system;
- Design the system architecture;
- Identify the principal system objectives;
- Develop design models;
- Specify object interfaces.

System context and interactions:

Understanding the relationships between the software that is being designed and its external environment is essential for deciding how to provide the required system functionality and how to structure the system to communicate with its environment. Understanding of the context also lets us establish the boundaries of the system. Setting the system boundaries helps us to decide what features are implemented in the system being designed and what features are in other associated systems.


A system context model is a structural model that ~~depends~~ demonstrates the other systems in the environment of the system being developed. An interaction model is a dynamic model that shows how the system interacts with its environment as it is used.


Use case Model: A use case model is a graphical representation of the interactions among the elements of a system and its external entities called actors. It consists of actors use cases and their relationships. The diagram is used to model system/sub-system of an application.

Purposes of use case diagram:

- ↳ Used together with requirement of a system.
- ↳ Used to get an outside view of system.
- ↳ Identify external and internal factors influencing the system.
- ↳ Focuses on functional requirement.

Symbols

1)  → External entity that interacts with the system. It can be human or machine, e.g., paypal, e-sewa etc.
<Role/Actor>

2)  → It is an action that the user performs within the system.
[Use case]
or action

- 3) [straight line] → Used to show interaction between the actor and user.
- 4) [Boundary] → System boundary used to represent the system scope.

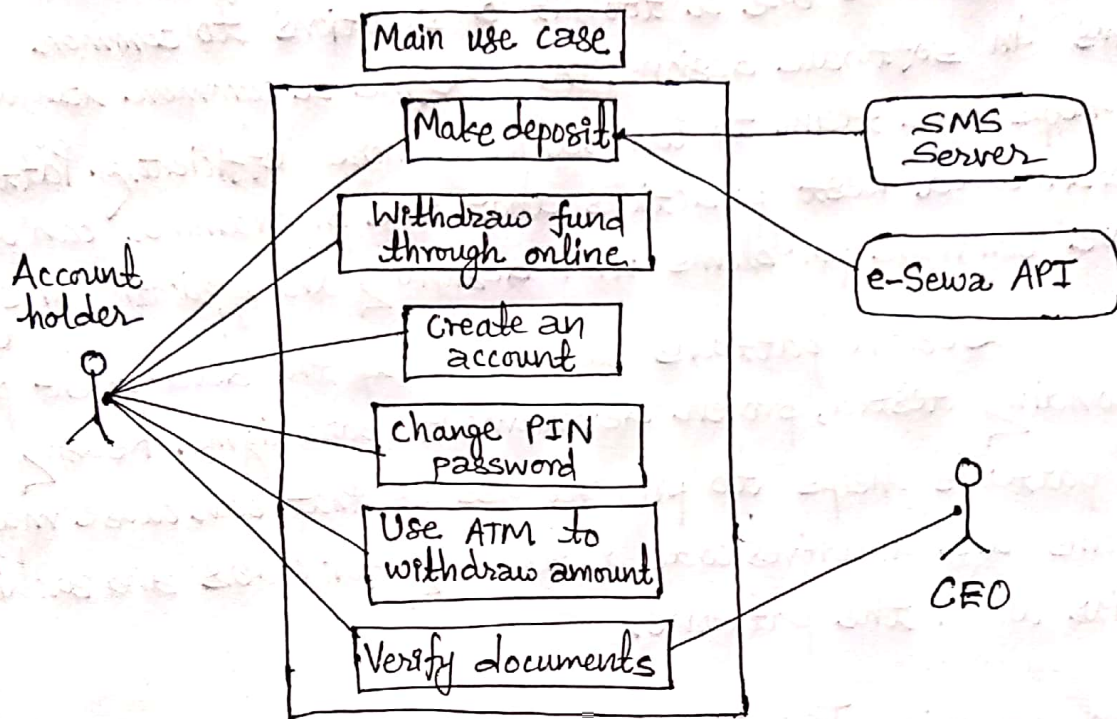


Fig: Bank ATM system use case diagram.

Architectural design: Architectural design is the process of decomposing the system into subsystems and establishing the relationship between subsystems. For this any of the architectural pattern is chosen from repository architecture, client server architecture or layered architecture.

Object class identification: Identifying object class is often a difficult part of object oriented design. There is no 'magic formula' for object identification. It relies on the skill, experience and domain knowledge of system designers. Object identification is an iterative process.

* Design Patterns:

Design patterns are typical solutions to common problems in software design. Each pattern is like a blueprint that we can customize to solve a particular design problem in our code. Patterns are a toolkit of solutions to common problems in software designs. They define a common language that helps our team to communicate more efficiently. Patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.

Design patterns can speed up the development process by providing tested, proven development paradigms. Reusing design patterns helps to prevent issues that can cause major problems and improves code readability for coders and architects familiar with the patterns.

* Implementation Issues:

Some aspect of implementation issues are:

1) Reuse: Nowadays, modern software are constructed by reusing existing components or systems. Software reuse is possible at different levels.

Object level: At this level, we directly reuse objects from a library rather than writing the code ourselves.

Component level: Components are collection of objects and object classes that we reuse in application systems. We can reuse the component by adding some code of our own.

System level: At this level, we reuse entire application systems.

2) Configuration Management: Configuration management is the name given to the general process of managing a changing software system. During the development process, we have to keep track of

the many different versions of each software component in a configuration management system. The aim of configuration management is to support the system integration process so that all developers can access the project code and documents in a controlled way, find out what changes have been made, and link components to create a system.

3) Host-target Development: Generally, the software is developed in one computer and executed on the same computer as the software development environment. Host target development is the methodology in which software is developed in one computer but runs on a separate computer.

⊗. Open Source Development:

Open source development refers to something that is free and that anyone can inspect, modify, share etc. Open source software is usually a free software product, where developers have access to the source code. They can enhance the program's performance, add some features, and fix errors. Open source development is an approach to software development in which the source code of a software system is published and volunteers are invited to participate in the development process. Examples of open source product are Linux, apache web server, MySQL, Java etc. A fundamental principle of open-source development is that source code should be freely available; this does not mean that anyone can do as they wish with that code. Developers involved should follow some terms and conditions and under some boundary conditions.