



# Unit 3:Process Overview

# PROCESS OVERVIEW

## Contents:

Development Stages

Development Life Cycle

# PROCESS OVERVIEW

Software development process provides a **basis** for the organized production of software, **using** a collection of **predefined techniques** and **notations**.

# Development Stages

▶ Software development has a **sequence of well defined stages**, each with a **distinct purpose**, **inputs** and **outputs**.

▶ The development stages can be summarized as:

1. System Conception
2. Analysis
3. System Design
4. Class Design
5. Implementation
6. Testing
7. Training
8. Deployment
9. Maintenance

.

# DEVELOPMENT STAGES...

## 1. System Conception:

System conception deals with the **genesis of application**

Initially somebody thinks of an idea for application, prepares a business case and sells the idea to organization.

The innovator must understand both business needs and technological capabilities

# CONT...

## 2. Analysis:

- Analysis focuses on creation of models, analyze, capture and scrutinize requirements by constructing models.
- During analysis, developers consider the available source of information and resolve ambiguities.

# DEVELOPMENT STAGES: ANALYSIS...

There are 2 sub stages :

1. Domain analysis.
2. Application analysis.


# 1. Domain Analysis

It focuses on **real world things** whose semantics the applications capture.

Eg: An airplane flight is a real world object that a **Flight Reservation system** must represent.

Domain objects exist independently of any applications and are meaningful to business experts.





Domain objects **carry information** about real world objects and are generally passive.

Domain analysis emphasizes **concepts and relationships**, with much of functionality being implicit in the class model

## 2. Application Analysis

Application analysis addresses the **computer aspects of application** that are visible to users.

Eg: A flight reservation screen is part of a flight.



Application objects **do not exist** in the **problem domain** and are meaningful only in the **context of applications**.

Application objects are not merely internal design decision because the users see them and must agree with them

### 3. SYSTEM DESIGN:

During system design, the developer makes **strategic decision** with board consequences.

You must **formulate an architecture** and **choose global strategies** and **policies** to guide the subsequently more detailed portion of design.



The architecture is the **high level design** strategy for solving application problem.

The **choice of architecture** is based on **requirements** as well as **past experiences**.

CONT...

The architecture must also support future modifications.

For large and complex problems their preparation must be interleaved.

Architecture helps to establish a model scope.

## 4. CLASS DESIGN

During class design, the developer expands and optimize analysis model. There is a shift in emphasis from application concepts towards complicate concepts.

Developers choose algorithm to implement major system functions.

## 5. IMPLEMENTATION

Implementation is the stage for writing the actual code.

Developers map design elements to programming language and database code.



## 6. TESTING

After implementation, the system is complete, but it must be **carefully tested** before **being commissioned** for actual use.

Testers once again **revisit** the original business requirements and **verify** that the system delivers the proper functionality.

Testers can test for errors(bugs) and portability issues.

Unit tests exercise small portions of code, they discover local problems.

System tests can discover broader portions meet specifications since they are applied to entire system

# CONT...


Both unit test and system test are necessary.

Testing must be planned from the beginning and many tests can be performed during implementation.

# 7. TRAINING

The organization must **train users** so that they can fully benefit from an application.

Training accelerates users on the **software learning curve**.



A separate team should prepare **user documentation** parallel to the development effort.

Quality control can then **check the software** against the **user documentation** to ensure that the **software meet its original goal**

## 8. DEPLOYMENT

The eventual system must work in field, on various platform and in various configuration.

Unexpected interactions can occur when a system is deployed in a customer environment.

Developers must tune the system under various loads and write scripts and install procedure.

## 9. MAINTENANCE

After the system is deployed, it must be maintained for continued success.

Bugs that appear during use must be resolved.

Enhancement requests must be considered.

# Development Life Cycle

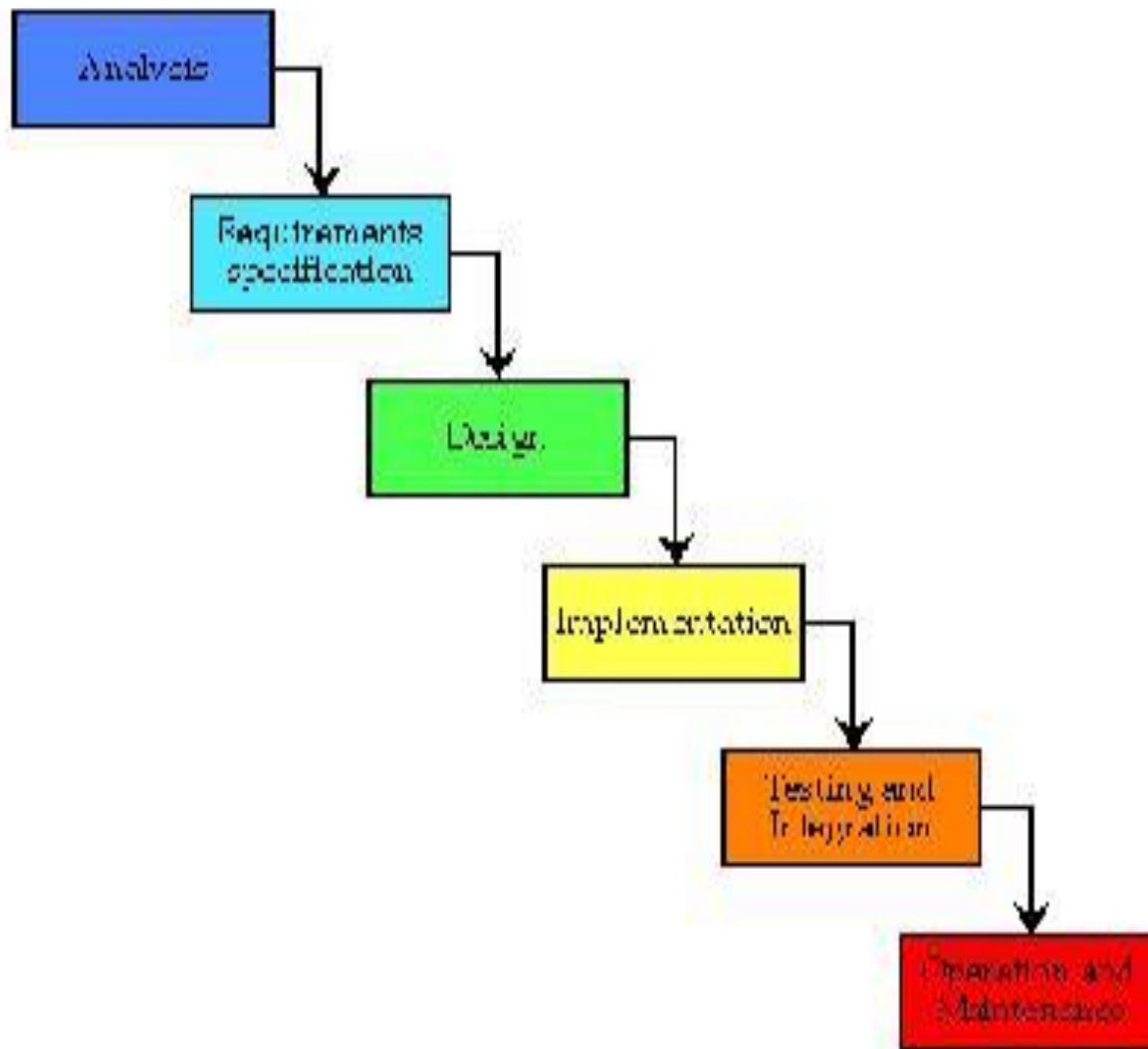
An object oriented approach to software development supports multiple life cycle style.




# 1. Waterfall Development

Waterfall approach dictates that developers perform the software development stages in a **rigid linear sequence** with **no backtracking**.

Developers **follow requirements**, then construct an **analysis model**, then perform a **system design**, then prepare a **class design**, followed by **implementation, testing and deployment**.





Each stage is completed entirely before the next stage is begun.

Waterfall approach is attempted by too many organizations when requirements are fluid.

# CONT...

This leads to a familiar situation where developers complain about inflexible software development.

Waterfall approach also does not delivers a useful system until completion.

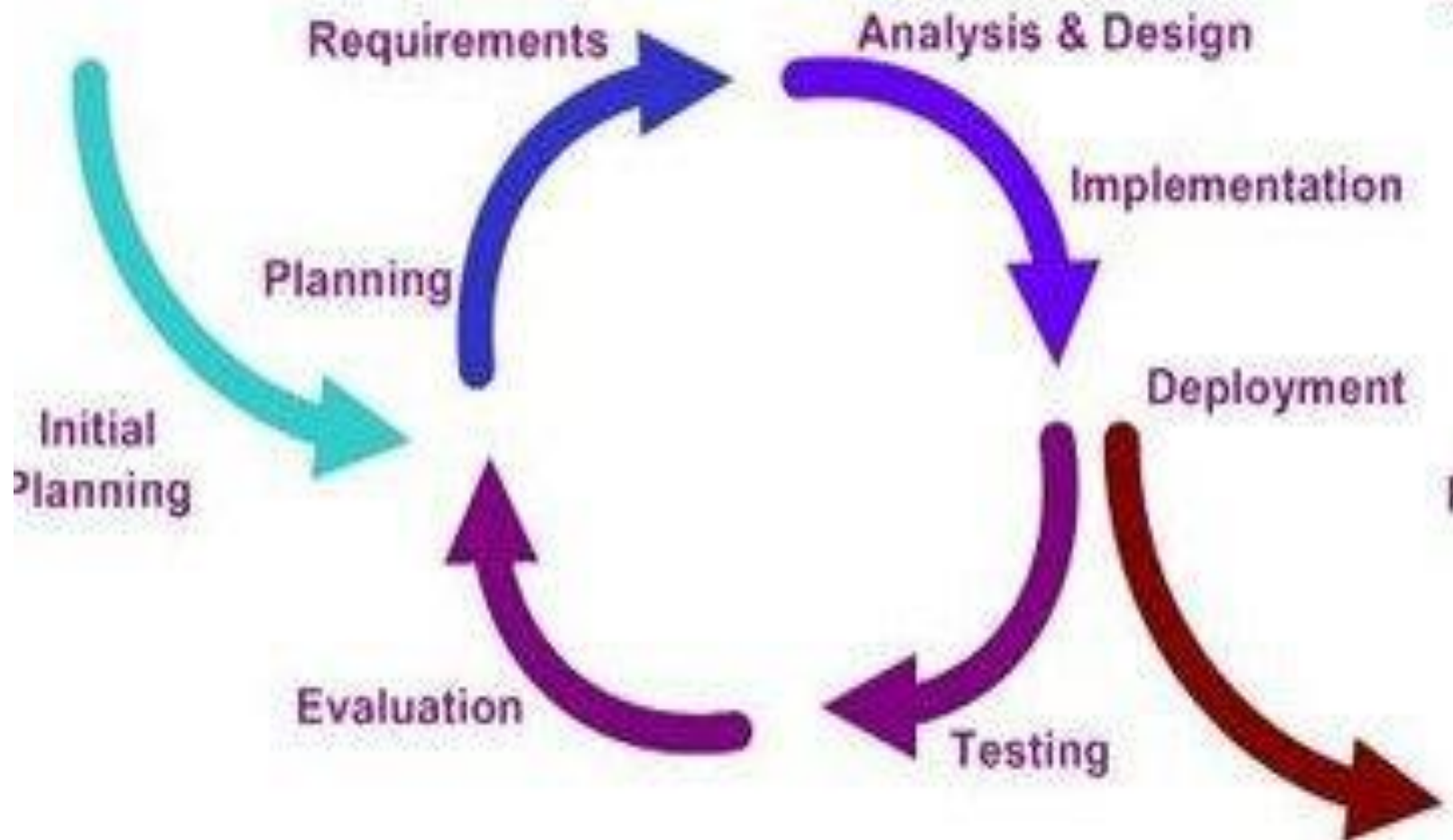
# Iterative Development

Iterative development is more flexible.

First you develop the nucleus of a system analyzing, designing, implementation and delivering working code.

Then you grow the scope of the system, adding properties and behavior to existing objects, as well as adding new kinds of objects.

There are multiple iterations as the system evolves to final deliverable.




# CONT...

The iteration includes a full complement of stages:

- **Analysis,**
- **design,**
- **implementation and**
- **testing.**

Iterative development can interleave the different stages and need not construct the entire system in lock step as in waterfall model.



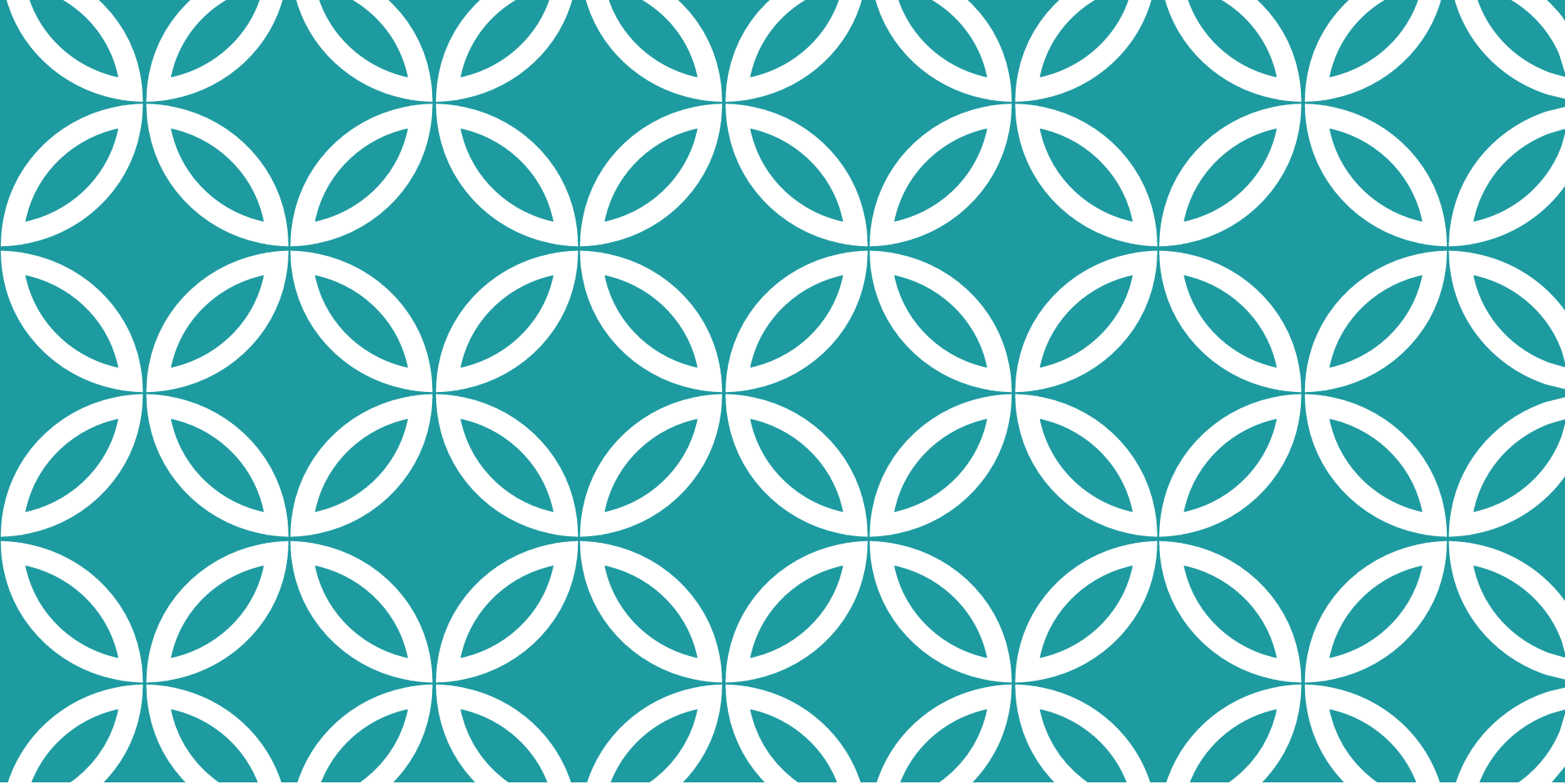
Each iteration ultimately **yields an executable system** that can be integrated and tested.

If there is **a problem**, you can **move backwards** to an earlier stages to **rework**.



# CONT...

Iterative development is the **best choice** for most applications because it gratefully **responds to change** and **minimize risk of failure**.



# SYSTEM CONCEPTION

# System Conception

System Conception deals with **genesis** of an application.

A person who understand both business needs and technology, thinks of an idea for an application.

Purpose of system conception is to **defer details** and **understand the big picture** –what need does the system meet.

# STEP-1: Devising A System Concept

Most ideas for new systems are extension of existing ideas.

Eg: HR dept may have a database of employee declaration for Tax and requires that a clerk enter these changes.

An obvious extension is to allow employees to view and enter their own changes.

Like wise, there are many issues to resolve (security, reliability, privacy...)

# SOME WAYS TO FIND **NEW SYSTEM CONCEPTS**:

1. **New Functionality:-** Add functionality to an existing system.
2. **Streamlining:-** Remove restrictions or generalize the way a system works.
3. **Simplification:-** Let ordinary persons perform tasks, which are previously assigned to specialists.
4. **Automation:-** Automate manual processes.
5. **Integration:-** Combine functionality from different systems.

CONT...

**6. Analogies:-** Look for analogies in other problem domains and see if they have useful ideas.

**7. Globalization:-** Travel to other countries and observe their cultural and business practices.

# STEP-2: Elaborating A Concept

A good system concept must **answer the following questions:**

1. Who is the applications for?
2. You should clearly understand which person and organization are stakeholders of the new system.

[the two most important kind of stakeholders are **financial sponsors** and the **end users**].

# CONT...

3. What problems will it solve?

4. Where will it be used?

5. Why is it needed?

6. How will it work?

.

.

.



For example: let us take up an ATM case study

Intended:

Develop software so that customer can access the bank's computers and carry out their own financial transactions without the mediation of a bank employee ( i.e, developing a netbanking application)

# STEP 3: Preparing A Problem Statement

Once the high level questions are answered, **requirement statements** that outlines the goals and **general approach** of the desired system can be **written**.

Throughout development, you should distinguish among requirements, design and implementation.

Requirements describe **how a system behaves** from user point of view.

Requirements and design decisions should not be mixed.

If you separate both you can present the **freedom to change a design**.

The system is considered as a black box

EG:

Some requirements for a **car** are:

When you press on accelerator pedal, the car goes faster and when you step on brake, the car slows down.

# CONT...

Design decision are **engineering choice** that provides the behavior specified by requirements.

Eg: Some design decisions are how the internal linkages are routed.

- How the engine is controlled.
- What kind of break pads are on the wheel.

Implementation deals with the ultimate realization in programming code.

# CONT...

A system concept document may include an implementation.

- Below table shows the problem statement:

Requirement statement	Design	Implementation
<ul style="list-style-type: none"><li>• Problem scope</li><li>• What is needed</li><li>• Application context</li><li>• Assumptions</li><li>• Performance needs</li></ul>	<ul style="list-style-type: none"><li>• General approach</li><li>• Algorithms</li><li>• Data structure</li><li>• Architecture</li><li>• Optimizations</li><li>• Capacity planning</li></ul>	<ul style="list-style-type: none"><li>• Platforms</li><li>• Hardware Specifications</li><li>• Software libraries</li><li>• Interface standards</li></ul>

# CONT...

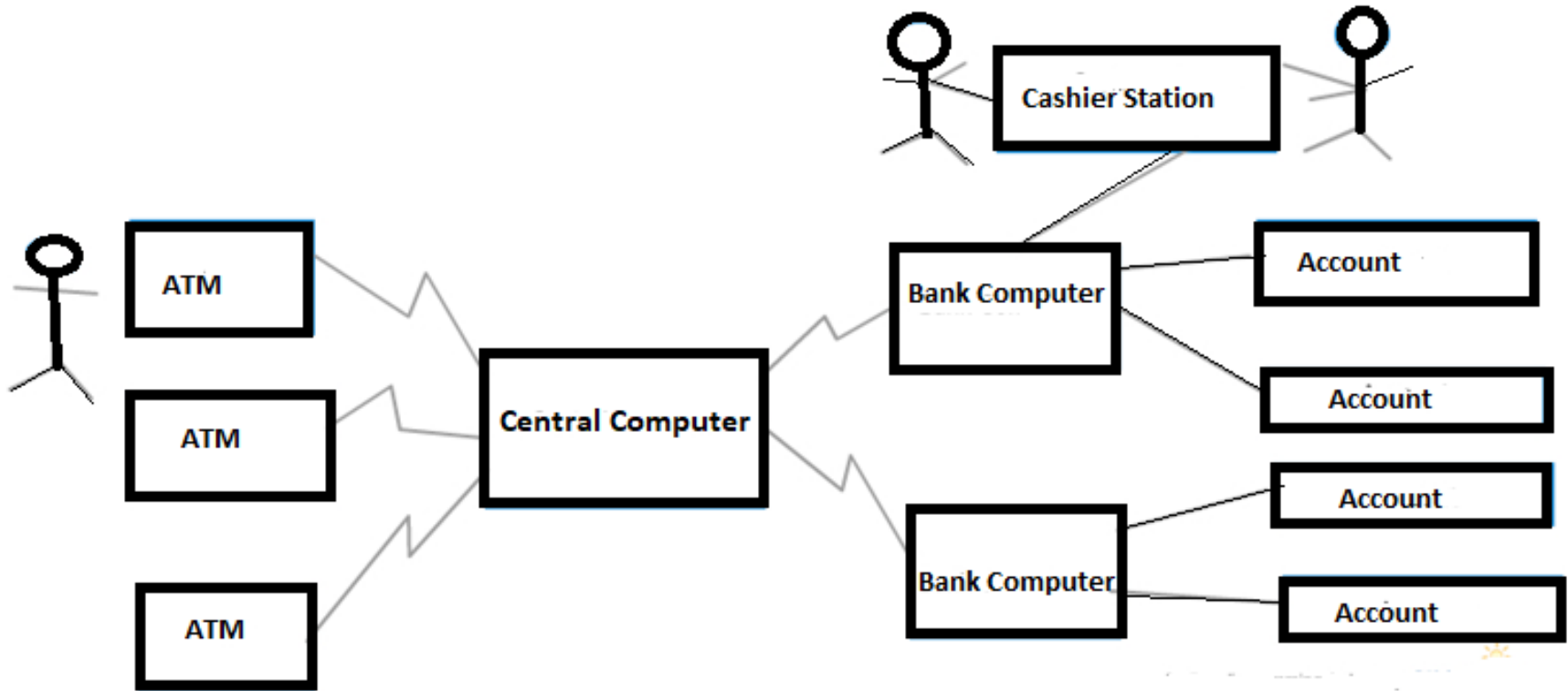
The problem statement **should state** what is to be done but **not** how it is to be implemented.

It should be **statements of needs**, not a proposal for system architecture.

The requestor should **avoid describing system** internal, as this restricts development flexibility.

The problem statement is just a starting point for understanding the problem, not a immutable document.

# ATM CASE STUDY...





Design the software to support a computerized banking network including both human cashier and ATM's to be shared by the consortium of banks.



## CONT...

Each bank provides its **own computer** to maintain its own accounts and process transactions against them.

Cashier stations are owned by individual banks and **communicates directly** with their own banks computers, human cashier's enter account and transaction data.

CONT...

ATM communicates with a **central computer** that clears transactions with the appropriate banks.

An ATM accepts cash card interacts with the user, communicates with central system to carry out the transaction, dispenses cash and prints receipts.

CONT...

The system requires appropriate record keeping and security provisions.

The system must handle concurrent access to the same account correctly.

## The ATM Case Study

- Who is the application for?
  - We are vendor building the software
- What problems will it solve?
  - Serve both bank and user
- Where will it be used?
  - Locations throughout the world
- When is it needed?
  - Revenue , investment
- Why is it needed?
  - Economic incentive. We have to demonstrate the techniques in the book
- How will it work
  - N-tier architecture, 3-tier architecture





END OF UNIT-3