UNIT I

INTRODUCTION

OBJECT ORIENTED SYSTEM DEVELOPMENT

- Software Development – Dynamic
- System Development – Information system solution
- Steps in System Development
  - Analysis
  - Design
  - Implementation
  - Testing
  - Maintenance

INTRODUCTION

- Software development Methodology
  - Development of an application design
  - Achieving goal based on the system requirements

INTRODUCTION

- Two orthogonal views of software:

  Program = Algorithm + Data Structure
- Traditional system development
  - Functionality/ Procedure
- Object oriented method
  - Data and functionality

Object Oriented System Development Methodology

- It is a way to develop software by building self-contained model or object that can be easily replaced, modified or reused
- OO environment
  - Collection of object
  - Encapsulation of data and functions (methods)

Object Oriented System Development Methodology

- Object
  - Grouped into classes
- OO
  - Based on objects
  - Each object is responsible for itself
  - (E.g.) window application – window object

    (that can open themselves on screen that can either display something or accept input)

Object Orientation

- Why?
  - High level of abstraction
  - Seamless transition among different phases of software development
  - Encouragement of goal programming technique
  - Promotion of reusability

Object Basics

- Object
  - A real world entity

  Can u give examples???

  - Well defined set of attributes in relation with other objects and set of things it can do

OO Philosophy

- Most programming languages provides programmer a way for describing the process
- In OOP it allows the base concepts of the language to be extended to include ideas and terms closer to those of its application
- Defines new data type with the existing primitive data type

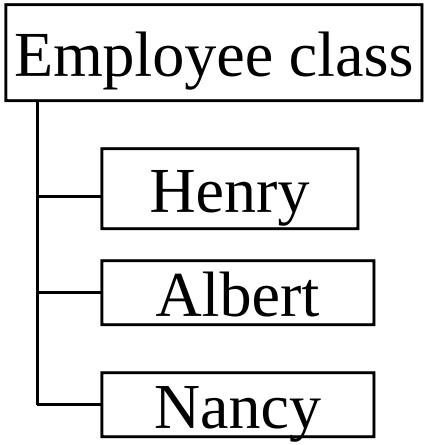
Objects and Classes

- Object - A combination of data and logic that represents some real world entity.
- It is natural to partition the world into objects, properties (state) and procedures (behavior)
- Class – used to distinguish one type of object from another
- In OOP it is a set of objects that share a common structure and behavior

Objects and Classes

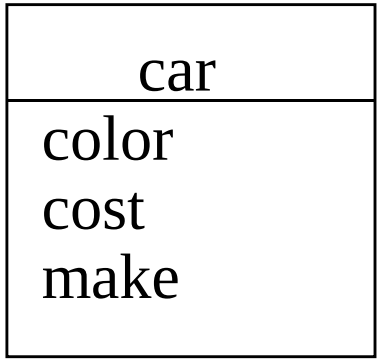
- Single object – instance of a class
- Class – are important mechanism for classifying object
- Method/Behavior – defined by its class

e.g.



Attributes and Behavior

- Attributes : Object’s state and properties
  - Properties represent the state of an object
- Behavior : set of things that object can do



object

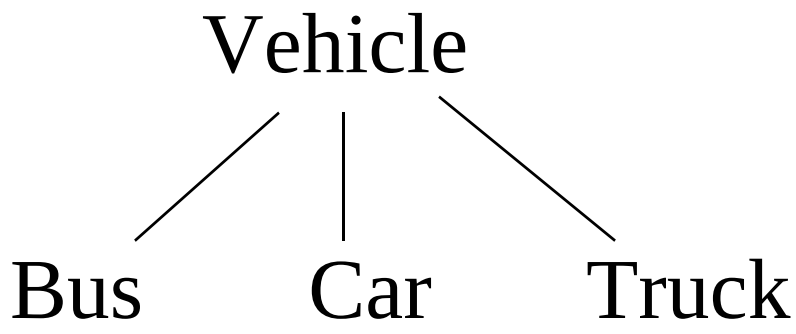
.

Encapsulation and information hiding

- Encapsulation : Binding or collection of data and functions within a single block
- Information Hiding : access specifiers
  - Public
  - Private
  - protected

Class Hierarchy

- Top most is the most general class
- Bottom most is most specific class
- Subclass, Super class
- Base class, Derived class

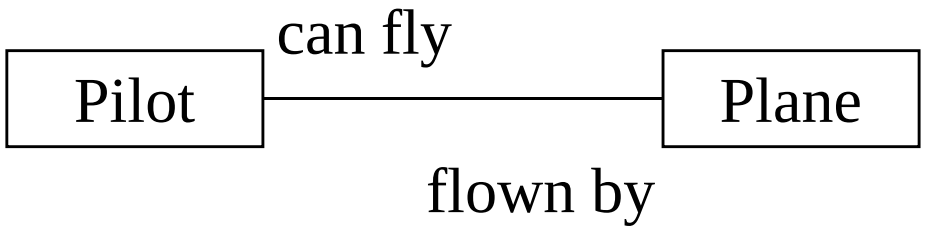


Inheritance and Polymorphism

- Inheritance – Reusability
- Dynamic Inheritance – add or delete object at run time
- Multiple Inheritance – derive a class from more than one base class
- Polymorphism – more than one form

Object relationships and Associations

- Association : representing relationship between objects or classes
- It is bidirectional
- E.g. a pilot can fly planes

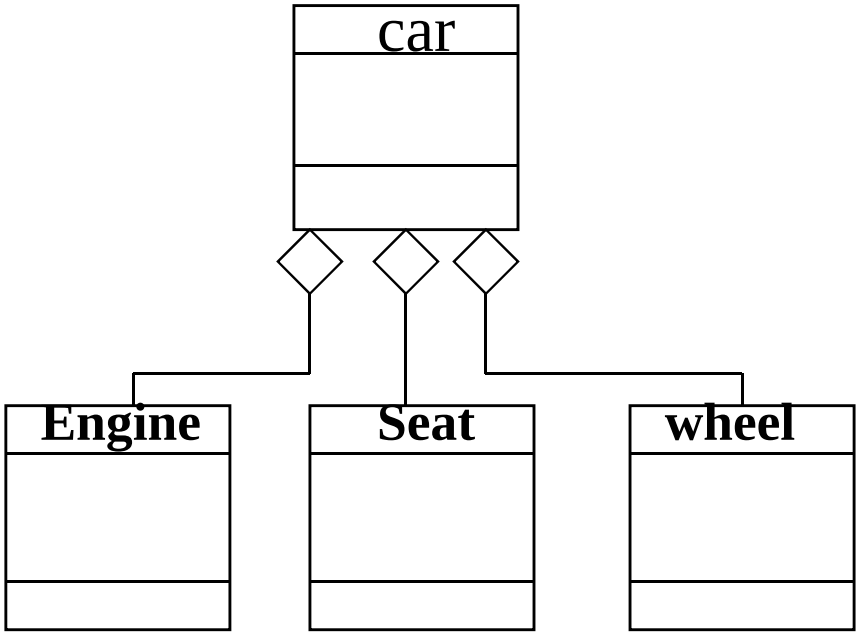


Associations

- Cardinality: how many time a class is associated with another class
- Consumer-Producer Association is a special form of association also called Client-Server association or a use relationship. We have one way interaction

Aggregation And Object Containment

- Each object has an identity
- One object can refer to other objects



Advanced Topics

- Object Identity
  - Feature of OO system – every object has its own identity
- Static and Dynamic binding
  - Process of determining which function to be invoked at run time is dynamic binding
    - Polymorphism (e.g. Cut operation)
  - Process of determination at compile time is static binding
    - Link section

Advanced Topics

- Object persistence:
  - Dealing with object life time
    - E.g. file/database
- Meta class:
  - Class is a set of object; all object should be defined within a class. Equivalent to defining a class belonging to another class is Meta class

Object Oriented System Development Life Cycle

- Software development process:
  - Consists of analysis, design, implementation, testing and refinement (to transfer user needs to software solution)
  - In other languages process can be started even if design is not completed
  - But in OO this cannot be done
  - Before implementing (coding) more time is spent for gathering requirements, developing requirement model and analysis model then into design model
  - After design model coding can be done quickly

The Software Development Process

- System Development is a generalized approach
- OO approach to software development is a specific approach
- System Development- viewed as a process itself
- It is a process of change, refinement, transformation or addition to existing product

The Software Development Process

- Within the process, it is possible to replace one sub process with a new one, as long as the new sub process has the same interface as the old one (inheritance)
- Process can be divided into small, interacting phases – sub process

The Software Development Process

- Each sub process must have the following:
  - Description in terms of how it works
  - Specification of input required for process
  - Specification of output to be produced

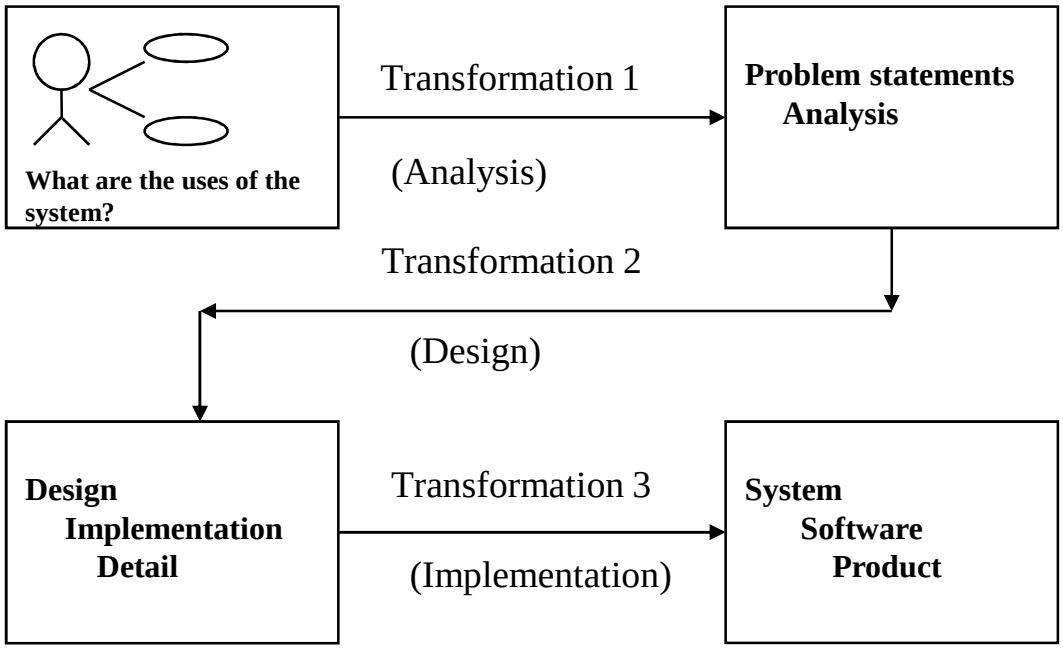
The Software Development Process

- In software development, process can be viewed as a series of transformation, in which output of one becomes the input of the subsequent transformation

The Software Development Process

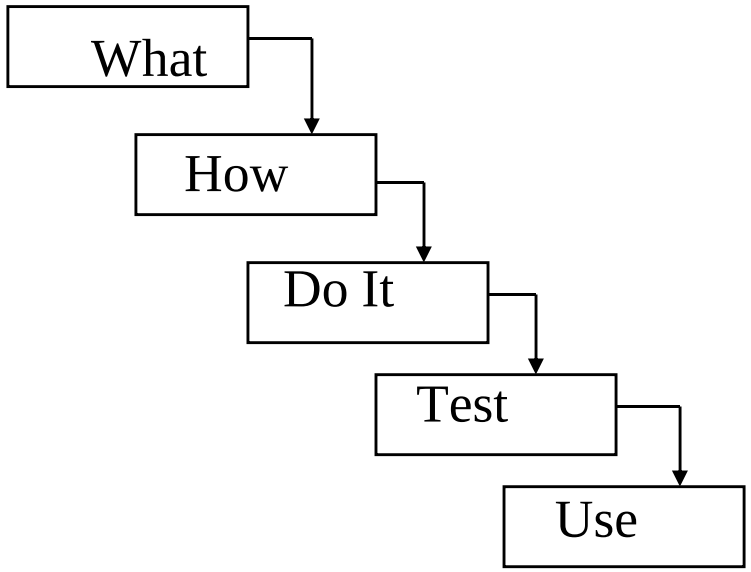
- Transformation 1(Analysis):
  - Translates user needs into system requirements and responsibilities
- Transformation 2 (Design):
  - Begins with problem statement and ends with detail design that can be transformed into an operational system
- Transformation 3 (Implementation):
  - Refines the detailed design into the system deployment that will satisfy the user’s needs

The Software Development Process



Waterfall Approach

- It is a simple example for software development process



Waterfall Approach

- This approach begins with what (requirements) is to be done
- After requirements have been determined, next it is to decide how (analysis) to accomplish them
- Next we Do It (Design & Implementation)
- Then test (Testing) the result and use (maintenance) the product

Building High Quality Software

- Software process transform users need via application domain to a software solution that satisfies those needs
- Once the system (program) exists, test it for free of bugs
- High quality product means all the process should be done before delivery of it to user

Building High Quality Software

- Goal of high quality product is user satisfaction
- To achieve high quality software, we need to be able to answer the following questions
  - How do we determine when the system is ready for delivery?
  - Is it now an operational system that satisfies users need?
  - Is it correct and operating as we thought it should?
  - Does it pass an evaluation process

Building High Quality Software

- There are basically two approach for system testing
  - How it has been built?
  - What it should do?

Four quality measures

- A system is evaluated in terms of four quality measures
  - Correctness
  - Verification
  - Correspondence
  - Validation

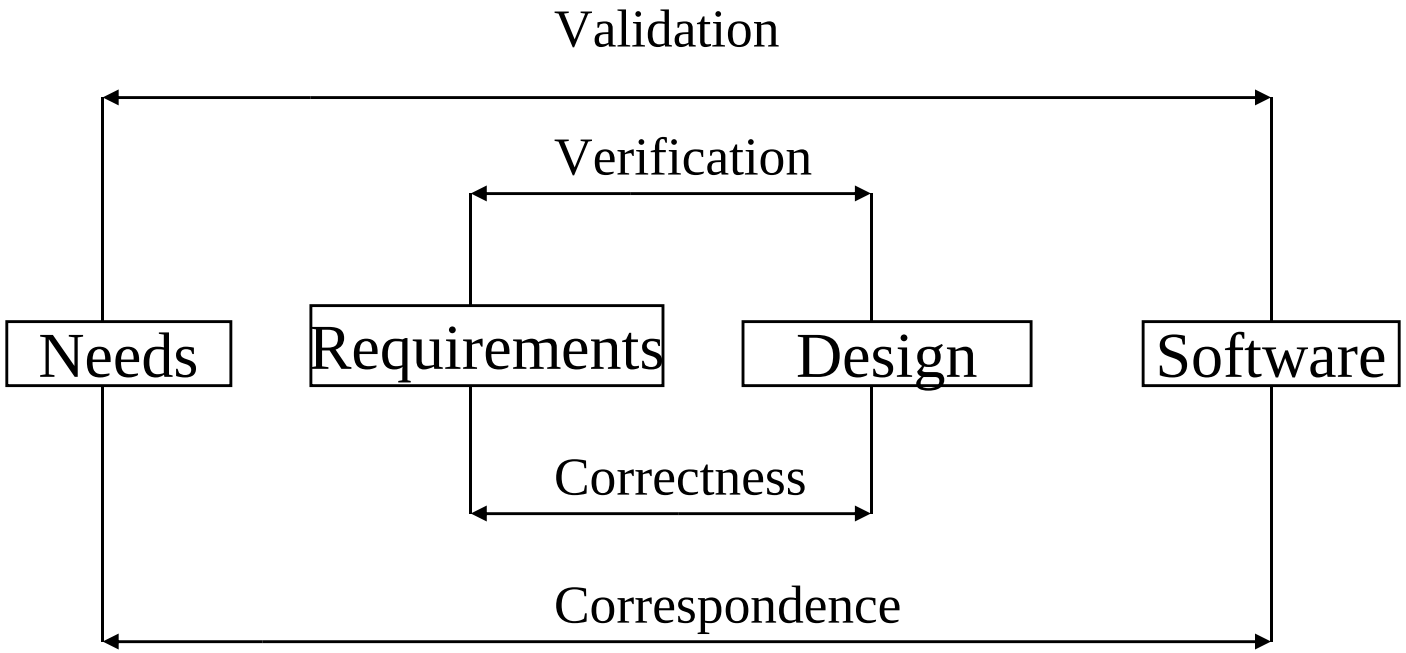
Four quality measures

- Correspondence
  - Measures how well the delivered system matches the needs of the operational environment as described in the original requirements statements
- Validation
  - It is the task of predicting correspondence. Correspondence cannot be predicted until the system is in place

Four quality measures

- Correctness
  - Measures the consistency of the product requirements with respect to the design specification
- Verification
  - It is the task of determining correctness

Four quality measures



Comparison of Verification and Validation

- Verification : Am I building the product right
- Validation : Am I building the right product
- Validation starts as soon as the project begins
- Verification begins only after a specification has been accepted
- Both are independent of each other

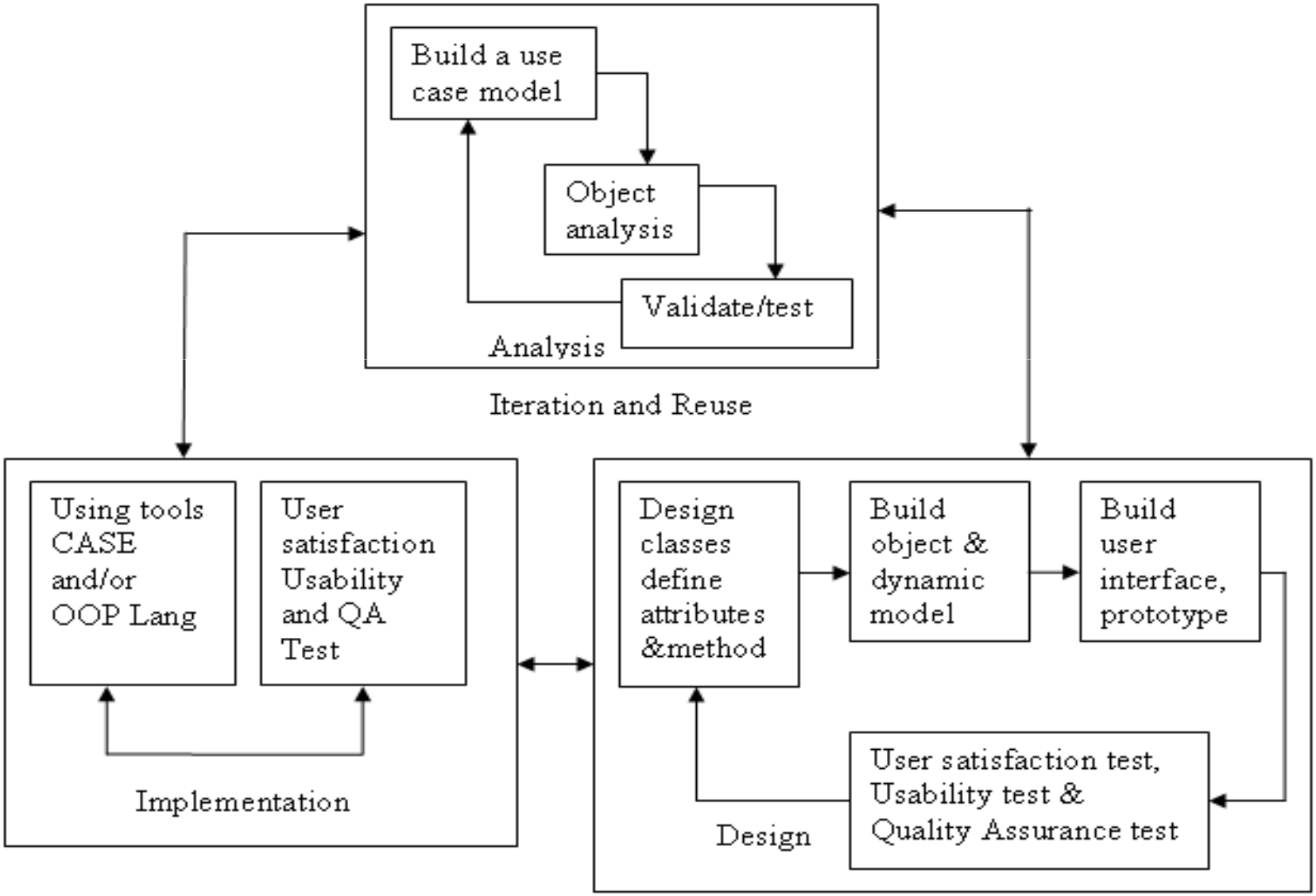
Object Oriented System Development : A Use Case Driven Approach

- OO Software Development Life Cycle (SDLC) consists of three macro process:
  - OO Analysis (OOA)
  - OO Design (OOD)
  - OO Implementation

Object Oriented System Development : A Use Case Driven Approach

- OO system development includes these activities:
  - OO Analysis
  - OO Design
  - Prototyping
  - Component Based Development (CBD) and Rapid Application Development
  - Incremental Testing

Object Oriented System Development : A Use Case Driven Approach



Object Oriented Analysis : Use case Driven

- This phase is concerned with determining the system requirements and identifying classes and their relationship to other classes in problem domain
- To understand system requirements we need to identify actors and users
- For this use-case concept is designed

Object Oriented Analysis : Use case Driven

- Use cases are scenarios
- Scenarios are a great way of examining who does what in the interaction among objects and what role they play i.e. interrelationship
- This intersection among object’s role to achieve a goal is called collaboration
- Use case modeling explains in detail about the different users and users needs. once use case identified then next sep is to identify classes and create their relationships

Object Oriented Design

- OOD is to design the classes identified during the analysis phase and the user interface
- In this phase addition classes, objects and relation may occur to fulfill the implementation requirements
- OOA and OOD are distinct disciplines but they can be intertwined

Object Oriented Design

- OO development is highly incremental
- First build the object model based on objects and their relationships, then iterate and refine the model
  - Design and refine classes
  - Design and refine attributes
  - Design and refine methods
  - Design and refine structures
  - Design and refine association

Object Oriented Design

- Guidelines to use OOD
  - Reuse rather than build a new class
  - Design a large no. of simple classes, rather a small no. of complex classes
  - Design methods
  - Critique what you have proposed. If possible, go back
  - Refine the classes

Prototyping

- Although OOA and OOD describe the system features, it is important to construct a prototype of some of the key system components shortly after the products are selected

Prototyping

- Prototype is a version of a software product developed in the early stages of the products life cycle for specific, experimental purposes
- It enables to understand how easy or difficult to implement some of the features of the system

Prototyping

- Provides the developer a means to test and refine the user interface and increase the usability of the system
- Categories of prototyping
  - Horizontal Prototype
  - Vertical Prototype
  - Analysis Prototype
  - Domain Prototype

Prototyping

- Horizontal Prototype
  - Simulation of the interface, but contains no functionality
  - Provides overall feel of the system and allows users to evaluate the interface on the basis of their normal, expected perception of the system

Prototyping

- Vertical Prototype
  - Is a subset of the system features with complete functionality
  - Advantage is few implemented functions can be tested in great depth
  - In practice prototypes are hybrid i.e. both horizontal and vertical

Prototyping

- Analysis Prototype
  - It is an aid for exploring the problem domain.
  - Used to inform the user and demonstrate the proof of a concept
- Domain Prototype
  - Is an aid for incremental development of the ultimate software solution

Prototyping

- Time required to produce a prototype is from few days to several weeks
- It depends on the type and function of the prototype
- Prototyping should involve representation from all user groups that will affected by the project

Prototyping

- Purpose of the review is:
  - To demonstrate that the prototype has been developed according to the specification
  - To collect information about errors and problems in the system
  - To give management and everyone connected with the project

Implementation
Component Based Development

- An industrial approach to software development
- Software components are built and tested in-house, using a wide range of technologies
- E.g. CASE Tools –allow their users to rapidly develop information systems

Component Based Development

- Goal of Case tools is automation of entire information system's development life cycle process using a set of integrated software tools, such as modeling, methodology and automatic code generation
- Code generation – only skeleton is generated modification has to be performed

Component Based Development

- CBD is a approach of application development where it moves from custom development to assembly of pre-built, pre-tested, reusable software component that operate with each other

Component Based Development

- Idea that underlies CBD is:
  - Application development can be improved significantly if application can be assembled quickly from pre-fabricated software component
  - An increasingly large collection of interpreted software components could be made available to developers in both general and specialist catalogs

Component Based Development

- CBD developer can assemble components to construct a complete software system
- Rewriting code from scratch is not possible
- For this a concept component wrapping technology can be used by which we can interact both the legacy and new software systems

Component Based Development

- **Software components:**
  - They are the functional units of a program, building blocks offering a collection of reusable services
  - They can request services from another component or deliver is own services on request

Component Based Development

- Components are independent, they work together to accomplish a task but don’t interfere with each other
- Each component is unaware of the inner work of the other component
- In short CBD is concerned with the implementation and system integration aspects of software development

Rapid Application Development

- It is a tool and technique that can be used to build an application faster than functional methods
- To achieve RAD, developer sacrifices the quality of the product for quicker delivery
- RAD is mainly concerned with "**Time to market**" than software development time

Rapid Application Development

- Task of RAD is to build the application quickly and incrementally implement the design and user requirements
- The main is to build a version of the application rapidly to see whether the problem have understood & determines whether the system does what it is supposed to do

Rapid Application Development

- RAD can have number of iterations
- So we can understand the problem and get improvement