UNIT – 2

Object Modelling Technique

Unit-02/Lecture-01

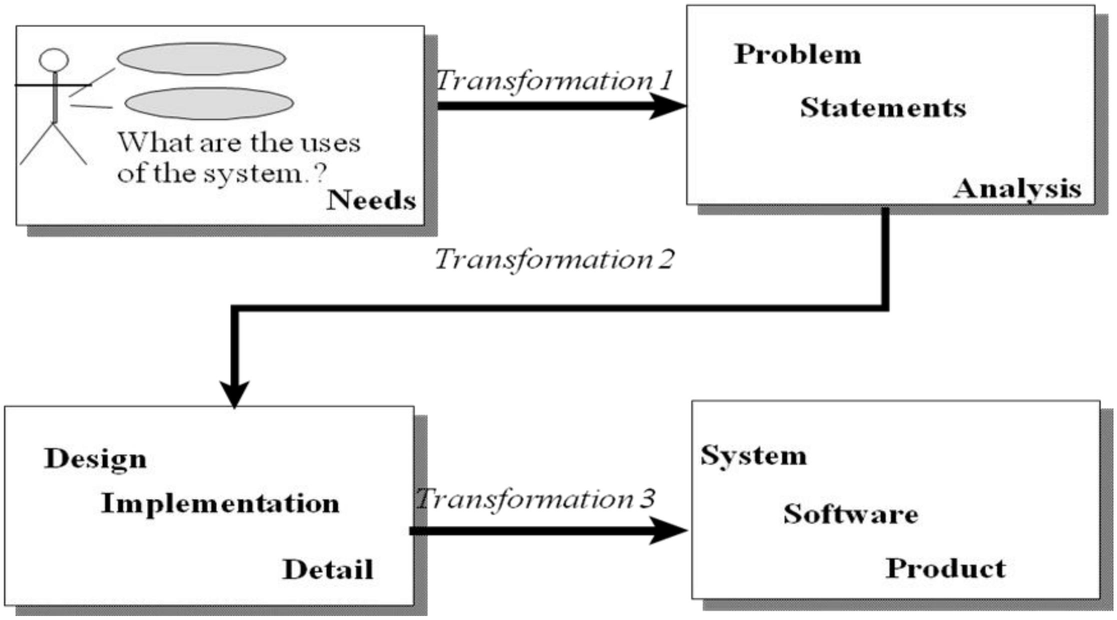
System design life cycle [RGPV /DEC-2010(10)]

Goals
The software development process Building high-quality software
Object-oriented systems development Use-case driven systems development Prototyping
Rapid application development Component-based development
Continuous testing and reusability

Software Process

The essence of the software process is the transformation of

 Users‘ needs to
 The application domain into
A software solution.



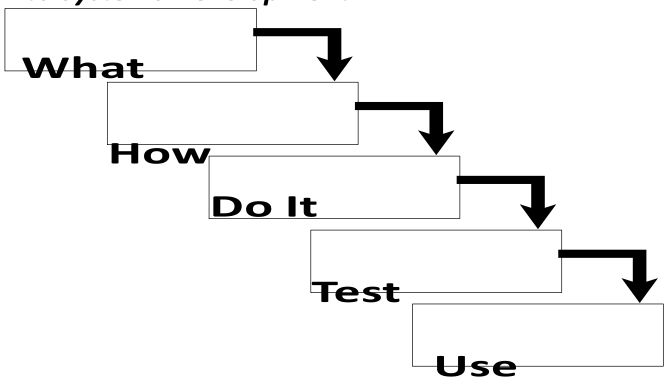
A software development process, also known as a software development life-cycle (SDLC), is a structure imposed on the development of a software product. Similar terms include software life cycle and software process. It is often considered a subset of systems development life cycle. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. Some people consider a life-cycle model a more general term and a software development process a more specific term. For example, there are many specific software development processes that 'fit' the spiral life-cycle model.

Software development organizations implement process methodologies to ease the process of development. Sometimes, contractors may require methodologies employed, an example is the U.S. defense industry, which requires a rating based on process models to obtain contracts.

The international standard for describing the method of selecting, implementing and monitoring the life cycle for software is ISO/IEC 12207.

A decades-long goal has been to find repeatable, predictable processes that improve productivity and quality. Some try to systematize or formalize the seemingly unruly task of designing software. Others apply project management techniques to designing software. Without effective project management, software projects can easily be delivered late or over budget. With large numbers of software projects not meeting their expectations in terms of functionality, cost, or delivery schedule, it is effective project management that appears to be lacking.

Traditional Waterfall Approach to Systems Development



Software Quality

- *There are two basic approaches to systems testing.*
- *We can test a system according to how it has been built.*
- *Alternatively, we can test the system with respect to what it should do.*

Quality Measures

- *Systems can be evaluated in terms of four quality measures:*

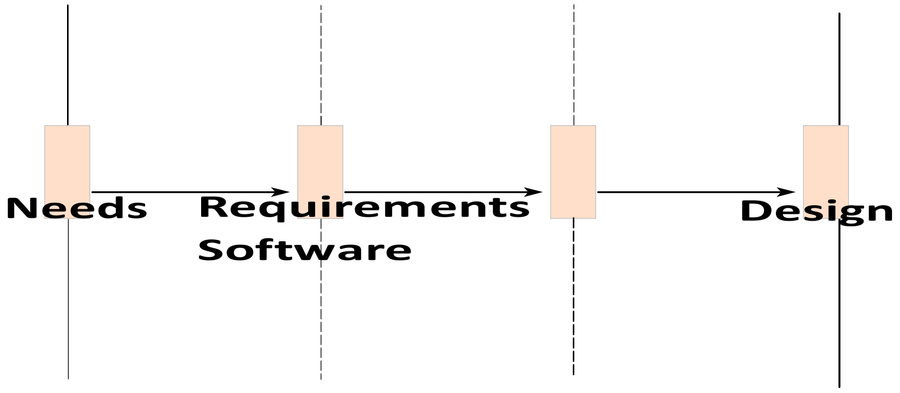
  –Correspondence
  – Correctness
  –Verification
  –Validation

- *Correspondence measures how well the delivered system corresponds to the needs of the operational environment.*

  It cannot be determined until the system is in place.

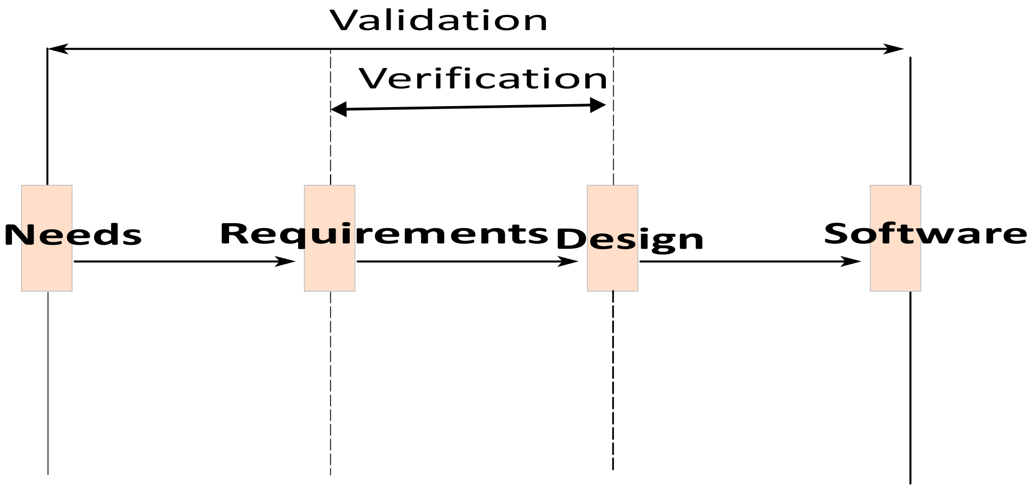
Correspondence

- Correctness measures the consistency of the product requirements with respect to the design specification.



Correctness

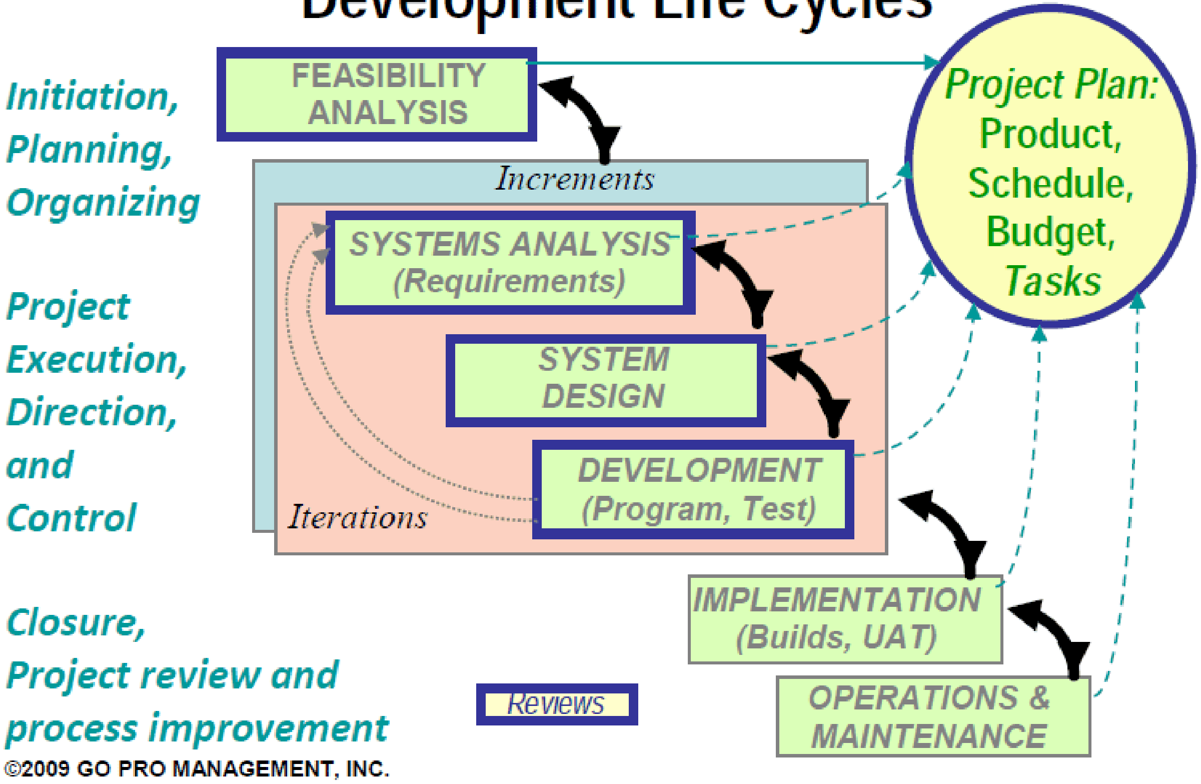
- *Verification* - "Am I building the product right?"
- *Validation* - "Am I building the right product?"



- Verification is to predict the correctness.
- Validation is to predict the correspondence.

S.NO	RGPV QUESTIONS	Year	Marks
	How do you develop an object oriented system development life cycle? Briefly discuss all the phases related to object oriented approach with an example.	Dec 2014	7
1.	How is object oriented software development different from a functional SDLC?	DEC,2010	10

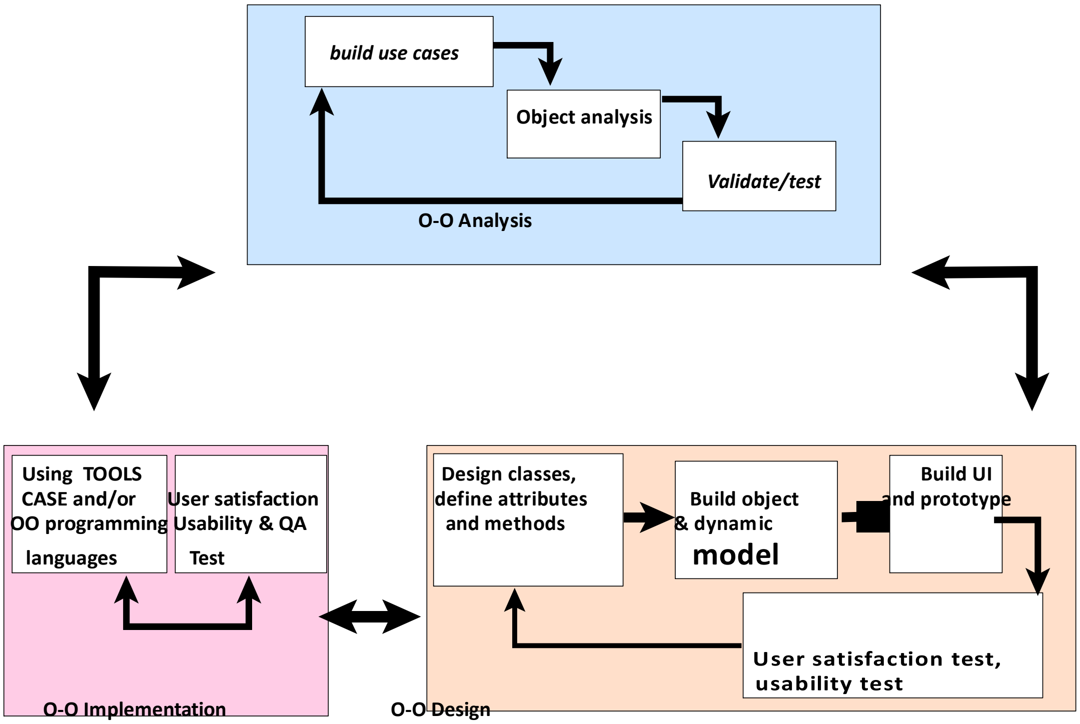
Project Management and System Development Life Cycles



©2009 GO PRO MANAGEMENT, INC.

Unit-02/Lecture-02

Object-Oriented Systems Development Approach **[RGPV /DEC-2010(10)]**

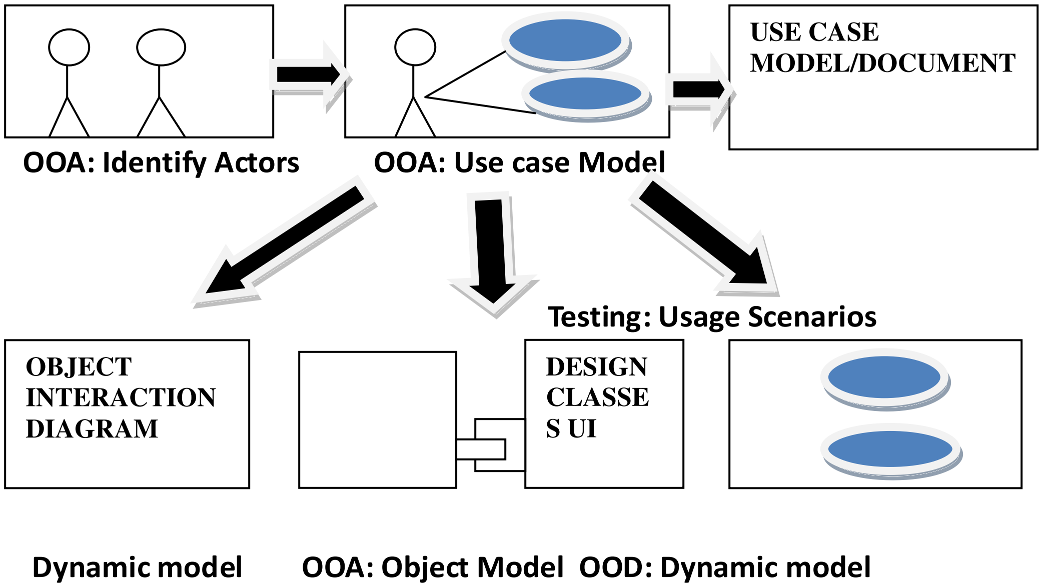

build use cases
Object analysis
Validate/test
O-O Analysis
Using TOOLS CASE and/or OO programming languages
User satisfaction Usability & QA Test
O-O Implementation
Design classes, define attributes and methods
Build object & dynamic model
Build UI and prototype
User satisfaction test, usability test
O-O Design

Object-Oriented Systems Development activities

- Object-oriented analysis.
- Object-oriented design.
- Prototyping.
- Component-based development.
- Incremental testing.

Use-case driven systems development

Use Case, is a name for a scenario to describe the user–computer system interaction.



Object-Oriented Analysis **[RGPV /June-2008(5)]**

OO analysis concerns with determining the system requirements and identifying classes and their relationships that make up an application.

Object-Oriented Design

•The goal of object-oriented design (OOD) is to design

•The classes identified during the analysis phase,

•The user interface and Data access

OOD activities include:

- Design and refine classes.
- Design and refine attributes.
- Design and refine methods.
- Design and refine structures.
- Design and refine associations.
  - Design User Interface or View layer classes.

○ Design data Access Layer classes.

Prototyping

o A Prototype enables you to fully understand how easy or difficult it will be to implement some of the features of the system.

o It can also give users a chance to comment on the usability and usefulness of the design.

Types of Prototypes

o A *horizontal prototype* is a simulation of the interface.

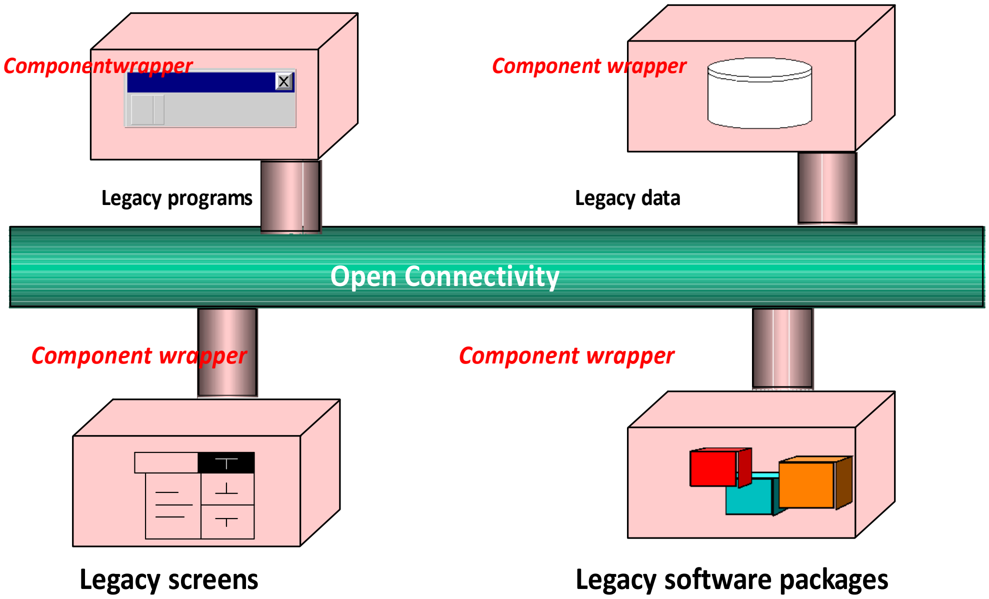
o A *vertical prototype* is a subset of the system features with complete functionality. o An *analysis prototype* is an aid for exploring the problem domain.

o A *domain prototype* is an aid for the incremental development of the ultimate software solution.

Component-based development (CBD)

• CBD is an industrialized approach to the software development process.

• Application development moves from custom development to assembly of pre-built, pre-tested, reusable software components that operate with each other.

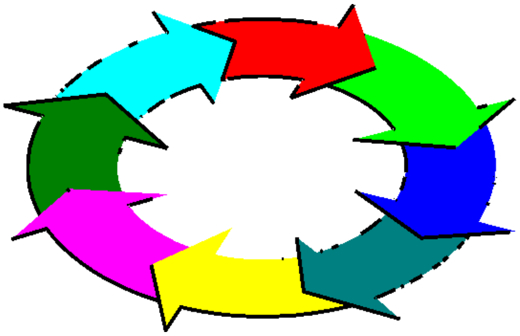


Rapid Application Development (RAD)

- RAD is a set of tools and techniques that can be used to build an application faster than typically possible with traditional methods.
- RAD does not replace SDLC but complements it, since it focuses more on process description and can be combined perfectly with the object-oriented approach.

Incremental Testing

- Software development and all of its activities including testing are an iterative process.
- If you wait until after development to test an application for bugs and performance, you could be wasting thousands of dollars and hours of time.



Reusability

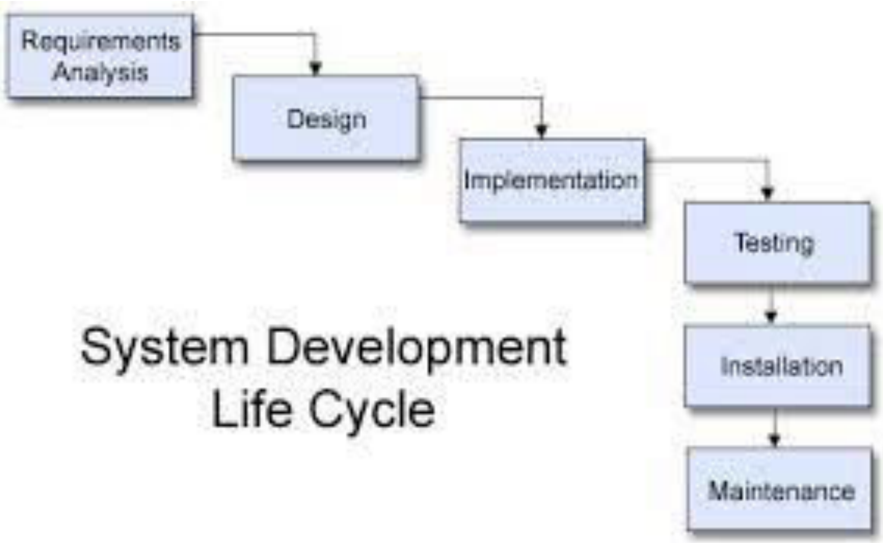
A major benefit of object-oriented systems development is reusability, and this is the most difficult promise to deliver on.

Reuse strategy

- Information hiding (encapsulation). • Conformance to naming standards.
- Creation and administration of an object repository.
- Encouragement by strategic management of reuse as opposed to constant redevelopment.
- Establishing targets for a percentage of the objects in the project to be reused (i.e., 50 percent reuse of objects).

The essence of the software process is the transformation of users' needs into a software solution. The O-O SDLC is an iterative process and is divided into analysis, design, prototyping/ implementation, and testing.

S.NO	RGPV QUESTIONS	Year	Marks
1.	Differentiate between analysis and design?	June 2008	5


Requirements Analysis
Design
Implementation
Testing
Installation
Maintenance
System Development Life Cycle

Object Oriented Analysis **[RGPV /DEC-2009(10)]**

Object-oriented analysis (OOA) looks at the problem domain, with the aim of producing a conceptual model of the information that exists in the area being analyzed. Analysis models do not consider any implementation constraints that might exist, such as concurrency, distribution, persistence, or how the system is to be built. Implementation constraints are dealt during object-oriented design (OOD). Analysis is done before the Design[*citation needed*].

The sources for the analysis can be a written requirements statement, a formal vision document, interviews with stakeholders or other interested parties. A system may be divided into multiple domains, representing different business, technological, or other areas of interest, each of which are analyzed separately.

The result of object-oriented analysis is a description of *what* the system is functionally required to do, in the form of a conceptual model. That will typically be presented as a set of use cases, one or more UML class diagrams, and a number of interaction diagrams. It may also include some kind of user interface mock-up. The purpose of object oriented analysis is to develop a model that describes computer software as it works to satisfy a set of customer defined requirements.

Object Oriented Analysis

1) a discovery process

2) clarifies and documents the requirements of a system

3) focuses on understanding the problem domain

4) discovers and documents the key problem domain classes

5) concerned with developing an object-oriented model of the problem domain

6) identified objects reflect the entities that are associated with the problem to be solved

OOA Definition

Definition

Object Oriented Analysis (OOA) is concerned with developing requirements and specifications expressed as an object model (population of interacting objects) of a system, as opposed to the traditional data or functional views.

Benefits

1) maintainability: simplified mapping to the real world a) less analysis effort
b) less complexity in system design c) easier verification by the user

2) reusability: reuse of the artifacts that are independent of the analysis method or programming language

3) productivity: direct mapping to the features implemented in Object Oriented Programming Languages

Object Modeling Technique (OMT): **[RGPV /June-2006(10)]**

OMT (Rumbaugh et al., 1991) was developed as an approach to software development. A fundamental assumption of OMT is that object-oriented thinking represents a more natural and intuitive way for people to reason about reality (ibid.:21), although this claim has been severely questioned, e.g. by Høydalsvik and Sindre, 1993; and Hanseth and Monteiro, 1994.

OMT is included here because Rumbaugh (1993:18) discusses enterprise modeling explicitly using OMT. OMT is also a widely popular and comprehensive approach that exemplifies the vast number of object-oriented approaches to modeling.

The purposes of modeling according to Rumbaugh et al. (1991:15) are

- *testing* physical entities before building them (simulation),
- *communication* with customers,
- *visualization* (alternative presentation of information), and
- reduction of *complexity*.

Hence, *understanding* and *simulation* is at the core. As a general modeling approach, OMT may be used to model all types of work. OMT proposes three main types of models:

- **Object model**

The object model represents the static and most stable phenomena in the modeled domain (Rumbaugh et al.,1991:21). Main concepts are *classes* and *associations*, with *attributes* and *operations*. *Aggregation* and *generalization* (with multiple inheritance) are predefined relationships.

- **Dynamic model**

The dynamic model represents a state/transition view on the model. Main concepts are *states*, *transitions* between states, and *events* to trigger transitions. *Actions* can be modeled as occurring within states. *Generalization* and *aggregation* (concur-rency) are predefined relationships.

- **Functional model**

The functional model handles the process perspective of the model, corresponding roughly to data flow diagrams. Main concepts are *process*, *data store*, *data flow*, and *actors*.

The entire OMT software development process has four phases: Analysis, system design, object design, and implementation of the software. Most of the modeling is performed in the analysis phase. The recommended method incorporates the following activities (Rumbaugh et al., 1991:261ff):

1. Develop a Problem Statement.
2. Build an Object Model:
    1. Identify object classes.
    2. Develop a data dictionary for classes, attributes, and associations.
    3. Add associations between classes.

    4. Add attributes for objects and links.
    5. Organize and simplify object classes using inheritance.
    6. Test access paths using scenarios and iterate the above steps as necessary.
    7. Group classes into modules, based on close coupling and related function.

3. Build a Dynamic Model:
    1. Prepare scenarios of typical interaction sequences.
    2. Identify events between objects and prepare an event trace for each scenario.
    3. Prepare an event flow diagram for the system.
    4. Develop a state diagram for each class that has important dynamic behavior.
    5. Check for consistency and completeness of events shared among the state diagrams.
4. Build a Functional Model:
    1. Identify input and output values.
    2. Use data flow diagrams as needed to show functional dependencies.
    3. Describe what each function does.
    4. Identify constraints.
    5. Specify optimization criteria.
5. Verify, iterate, and refine the three models:
    1. Add most important operations to the object model.
    2. Verify that classes, associations, attributes and operations are consistent and complete, check with problem statement.
    3. Iterate steps to complete the analysis.

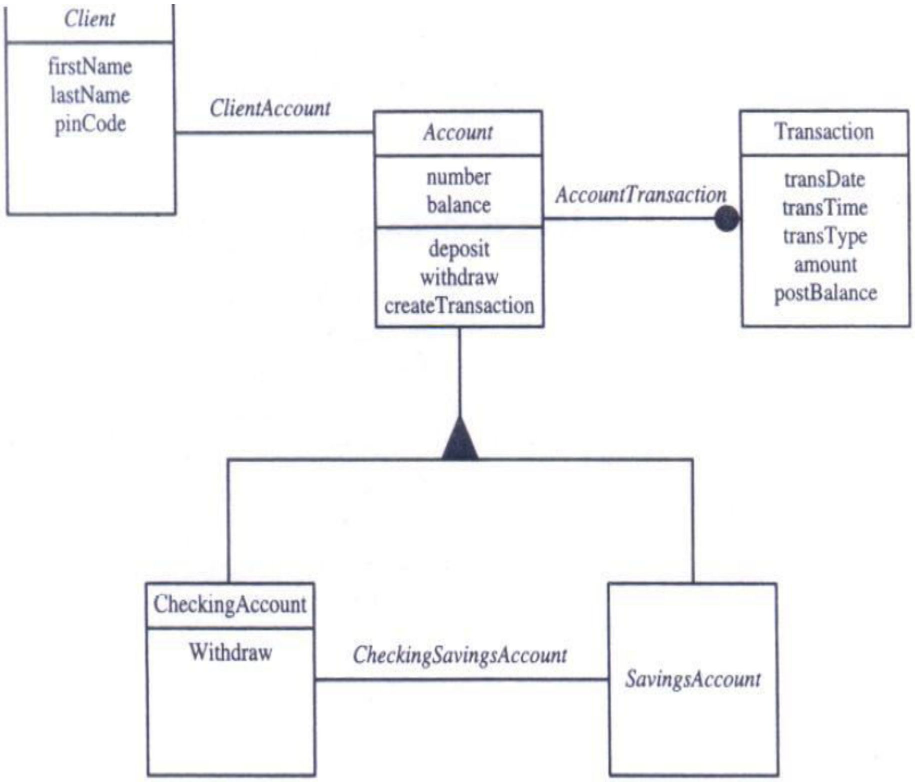
S.NO	RGPV QUESTIONS	Year	Marks
1.	How will you define the object oriented analysis (OOA)approach? What are the different OOA Methods?	Dec,2009	10
2.	Describe object modelling techniques(OMT) METHODOLOGY	JUNE,2006	10

object model **[RGPV /DEC-2008(10),JUNE-2009(10)]**

The object model describes the structure of objects in a system: their identity, relationships to other objects, attributes, and operations. The object model is represented graphically with an object diagram (see Fig). The object diagram contains classes interconnected by association lines. Each class represents a set of individual objects. The association lines establish relationships among the classes. Each association line represents a set of links from the objects of one class to the objects of another class.

DYNAMIC MODEL

OMT provides a detailed and comprehensive dynamic model, in addition to letting you depict states, transitions, events, and actions. The OMT state transition diagram is a network of states and events (see Fig). Each state receives one or more events, at which time it makes the transition to the next state. The next state depends on the current state as well as the events.



The OMT object model of a bank system. The boxes represent classes and the filled triangle represents specialization. Association between Account and transaction is one too many; since one account can have many transactions, the filled circle represents many (zero or more). The relationship between Client and Account classes is one to one: A client can have only one account and account can belong to only one person (in this model joint accounts are not allowed).

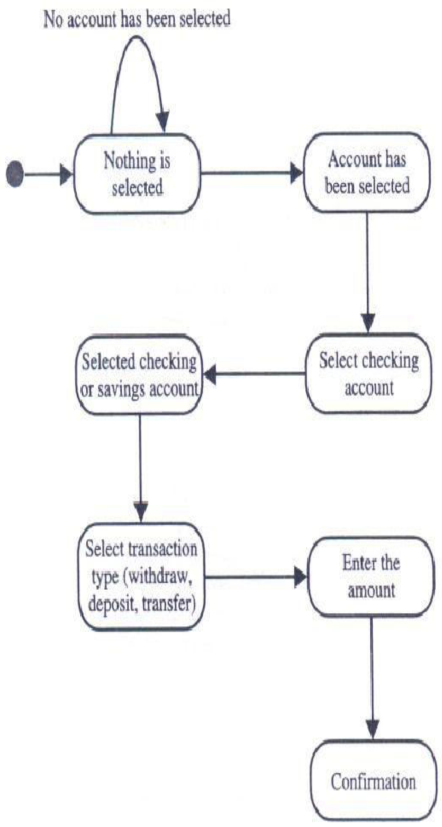
function model,

The OMT data flow diagram (DFD) shows the flow of data between different processes in a business. An OMT DFD provides a simple and intuitive method for describing business processes without focusing on the details of computer systems.

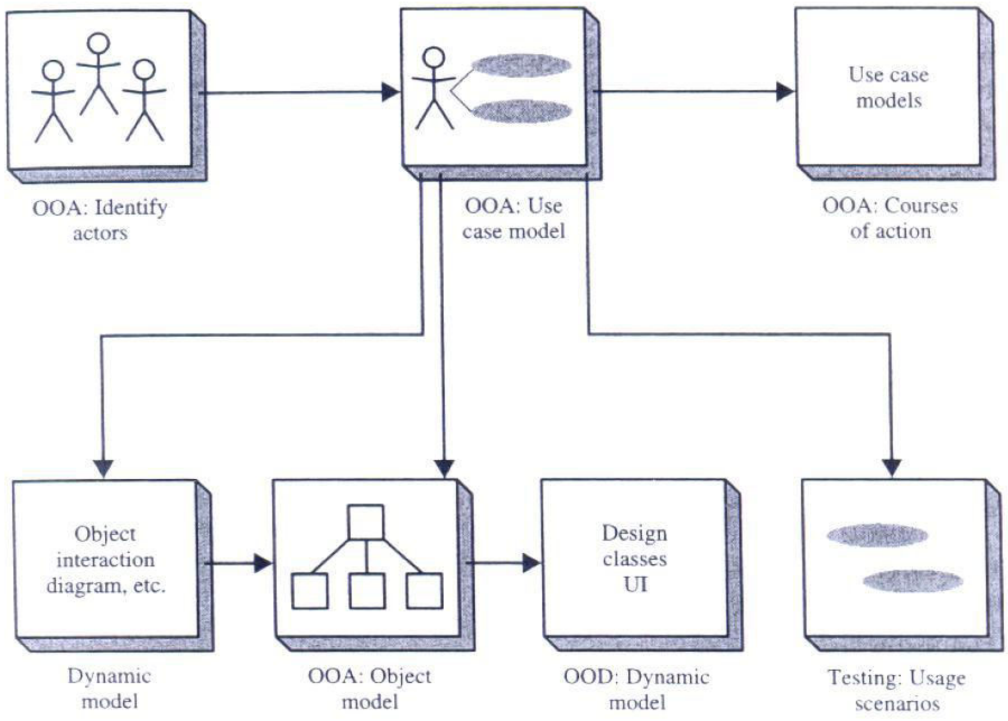
Data flow diagrams use four primary symbols:
1. The *process* is any function being performed; for example, verify Password or PIN in the ATM system (see Fig).
2. The *data flow* shows the direction of data element movement; for example, PIN code. 3. The *data store* is a location where data are stored; for example, account is a data store in the ATM example.
4. An *external entity* is a source or destination of a data element; for example, the ATM card reader.

Overall, the Rumbaugh et al. OMT methodology provides one of the strongest tool sets for the analysis and design of object-oriented systems

State transition diagram for the bank application user interface. The round boxes represent states and the arrows represent transitions.


No account has been selected
Nothing is selected
Account has been selected
Select checking account
Selected checking or savings account
Select transaction type (withdraw, deposit, transfer)
Enter the amount
Confirmation

relationship among models,



S.NO	RGPV QUESTIONS	Year	Marks
1.	Explain the following: object model, dynamic model, functional model, relationship between different model.	Dec.2008	10
2.	What is dynamic model? Illustrate with the help of state transition diagram?	June,2009	10

Unit-02/Lecture-05

object diagrams **[RGPV /DEC-2013(7),JUNE-2006(5)]**

Object Diagrams:

1) show a set of objects and their relationships
2) static snapshots of element instances found in class diagrams

Object Diagram

1) models the instances of classes contained in class diagrams

2) shows a set of objects and their relationships at one time

3) modelling object structures involves taking a snapshot of a system at a given moment in time

4) is an instance of a class diagram or the static part of an interaction diagram

5) it contains objects and links

Object Diagram Usage

Object diagrams are used to:

1) visualize
2) specify
3) construct
4) document

The existence of certain instances in a system, together with their relationships.

Creating an Object Diagram

1) identify the function/behaviour of interest that results from interaction of classes, interfaces and other artifacts

2) for each function/behaviour, identify the artifacts that participate in the collaboration as well as their relationships

3) consider one scenario that invokes the function/behaviour, freeze the scenario and render each participating object

4) expose the state and attribute values of each object, as necessary to understand the scenario

5) expose the links among these objects

Object Diagram Example

The following diagram is an example of an object diagram. It represents the *Order management system* which we have discussed in *Class Diagram*. The following diagram is an instance of the system at a particular time of purchase. It has the following object

- Customer
- Order
- SpecialOrder
- NormalOrder

Now the customer object (C) is associated with three order objects (O1, O2 and O3). These order objects are associated with special order and normal order objects (S1, S2 and N1). The customer is having the following three orders with different numbers (12, 32 and 40) for the particular time considered.

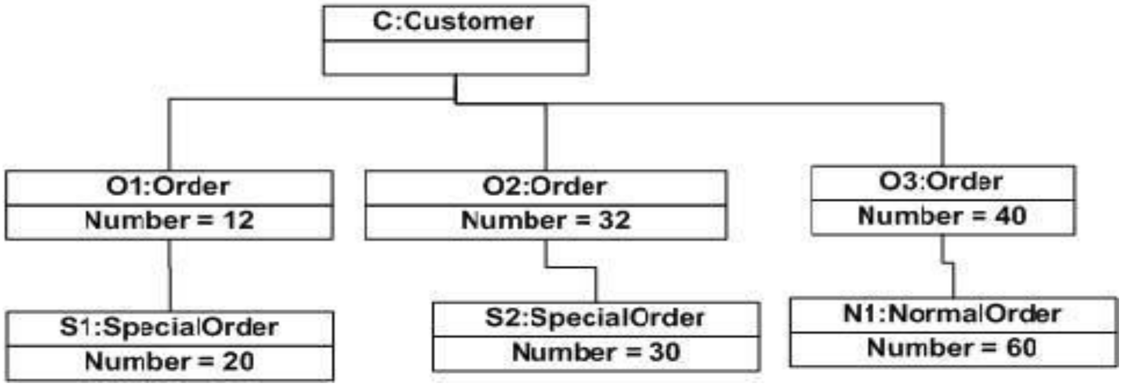
Now the customer can increase number of orders in future and in that scenario the object diagram will reflect that. If order, special order and normal order objects are observed then we you will find that they are having some values.

For orders the values are 12, 32, and 40 which implies that the objects are having these values for the particular moment (here the particular time when the purchase is made is considered as the moment) when the instance is captured.

The same is for special order and normal order objects which are having number of orders as 20, 30 and 60. If a different time of purchase is considered then these values will change accordingly.

So the following object diagram has been drawn considering all the points mentioned above

Object diagram of an order management system



S.NO	RGPV QUESTIONS	Year	Marks
1.	What is object diagram?	Dec,2013	7
2.	Draw and explain the notation of object diagram	June,2006	5

State diagrams [RGPV /DEC-2009(5)]

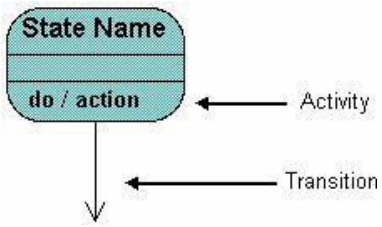
State diagrams are used to describe the behavior of a system. State diagrams describe all of the possible states of an object as events occur. Each diagram usually represents objects of a single class and track the different states of its objects through the system.

When to Use: State Diagrams

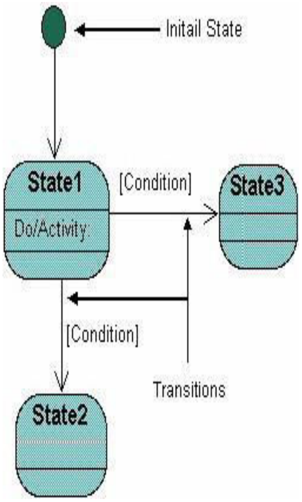
Use state diagrams to demonstrate the behavior of an object through many use cases of the system. Only use state diagrams for classes where it is necessary to understand the behavior of the object through the entire system. Not all classes will require a state diagram and state diagrams are not useful for describing the collaboration of all objects in a use case. State diagrams are other combined with other diagrams such as interaction diagrams and activity diagrams.

How to Draw: State Diagrams

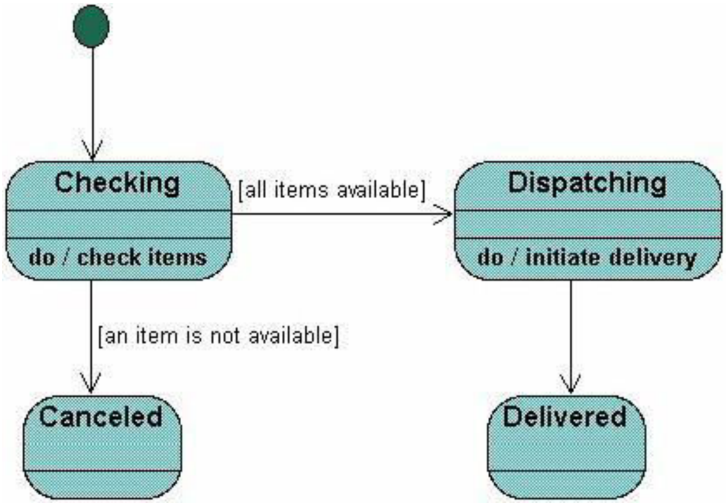
State diagrams have very few elements. The basic elements are rounded boxes representing the state of the object and arrows indicting the transition to the next state. The activity section of the state symbol depicts what activities the object will be doing while it is in that state.



All state diagrams being with an initial state of the object. This is the state of the object when it is created. After the initial state the object begins changing states. Conditions based on the activities can determine what the next state the object transitions to.

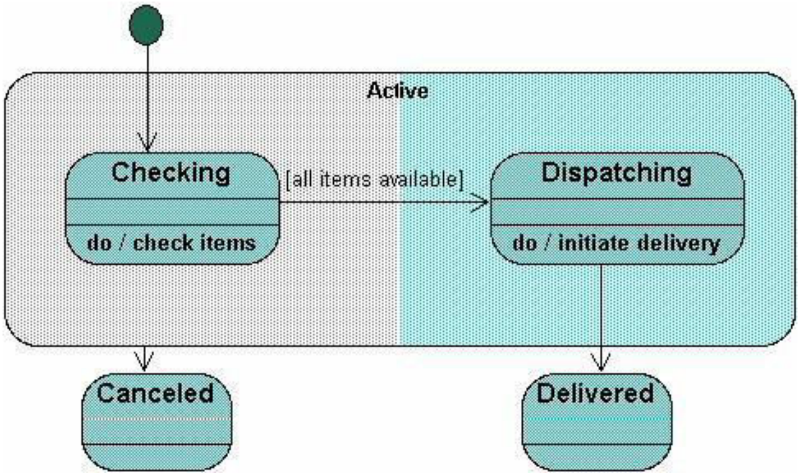


Below is an example of a state diagram might look like for an Order object. When the object enters the Checking state it performs the activity "check items." After the activity is completed the object transitions to the next state based on the conditions [all items available] or [an item is not available]. If an item is not available the order is canceled. If all items are available then the order is dispatched. When the object transitions to the Dispatching state the activity "initiate delivery" is performed. After this activity is complete the object transitions again to the Delivered state



State diagrams can also show a super-state for the object. A super-state is used when many transitions lead to the a certain state. Instead of showing all of the transitions from each state to the redundant state a super-state can be used to show that all of the states inside of the super-state can transition to the redundant state. This helps make the state diagram easier to read.

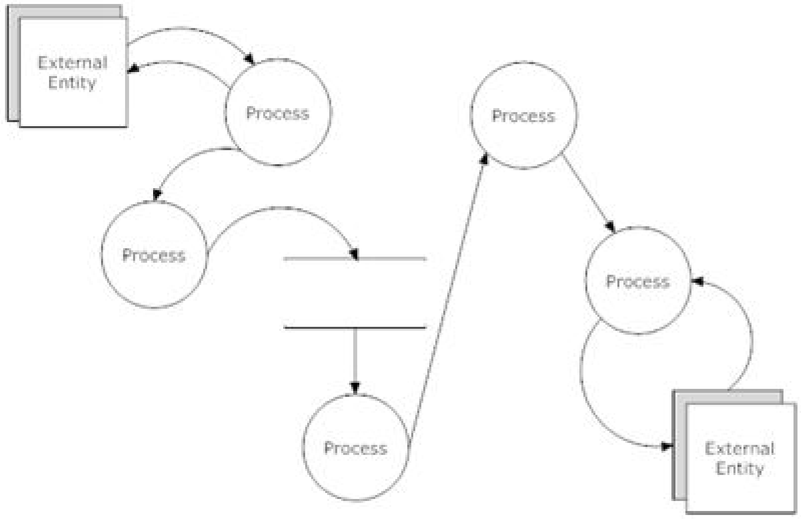
The diagram below shows a super-state. Both the Checking and Dispatching states can transition into the Canceled state, so a transition is shown from a super-state named Active to the state Cancel. By contrast, the state Dispatching can only transition to the Delivered state, so we show an arrow only from the Dispatching state to the Delivered state.



S.NO	RGPV QUESTION	YEAR	MARKS
1.	Explain state diagram with suitable example	Dec,2009	5

Data flow diagrams [RGPV /DEC-2010(10)]

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs.



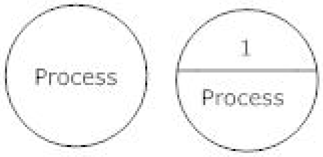
Data Flow Diagram Notations

You can use two different types of notations on your data flow diagrams: *Yourdon & Coad* or *Gane & Sarson*.

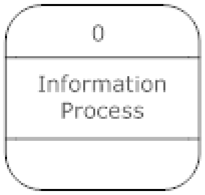
Process Notations

Process

A process transforms incoming data flow into outgoing data flow.



Yourdon and Coad Process Notations

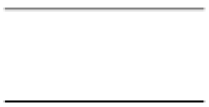


Gane and Sarson Process Notation

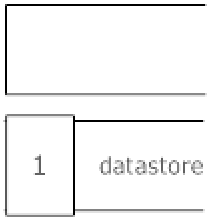
Datastore Notations

DataStore

Datastores are repositories of data in the system. They are sometimes also referred to as files.



Yourdon and Coad Datastore Notations

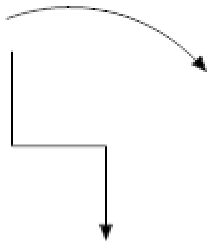


Gane and Sarson Datastore Notations

Dataflow Notations

Dataflow

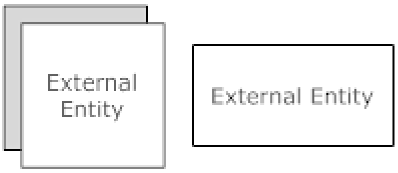
Dataflows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.



External Entity Notations

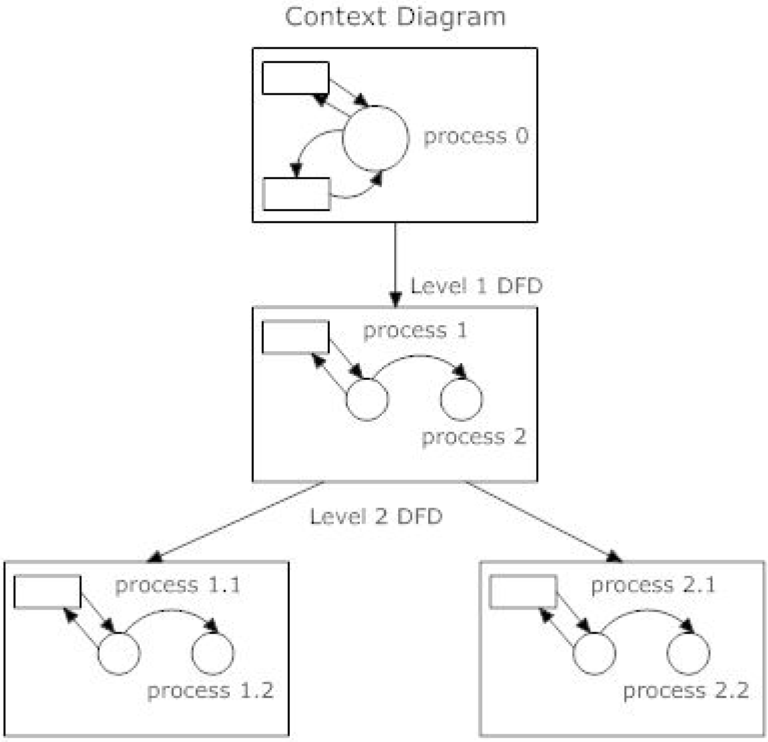
External Entity

External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the system's inputs and outputs.



Data Flow Diagram Layers

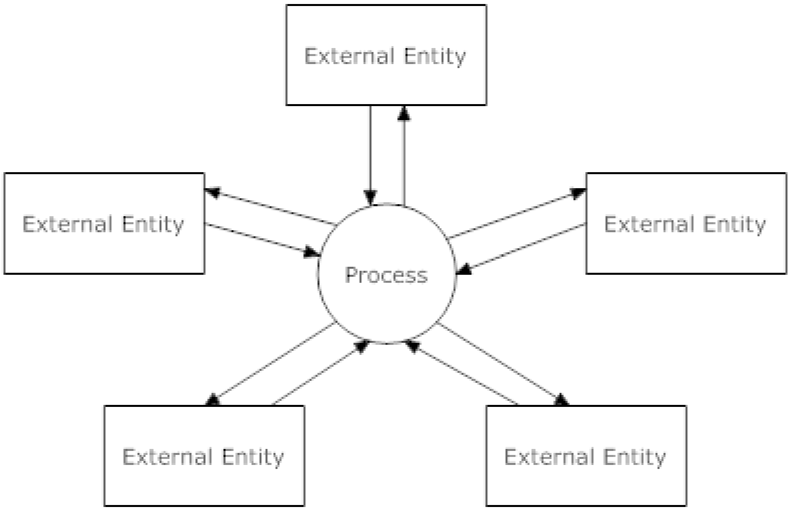
Draw data flow diagrams in several nested layers. A single process node on a high level diagram can be expanded to show a more detailed data flow diagram. Draw the context diagram first, followed by various layers of data flow diagrams.



The nesting of data flow layers

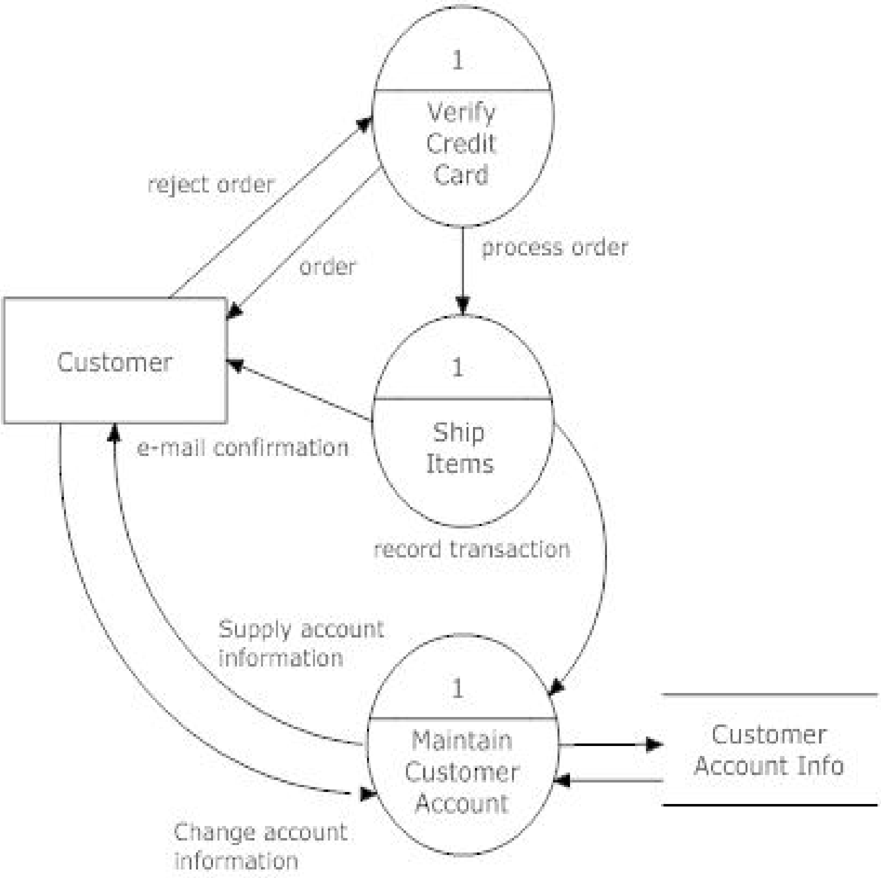
Context Diagrams

A context diagram is a top level (also known as Level 0) data flow diagram. It only contains one process node (process 0) that generalizes the function of the entire system in relationship to external entities.



DFD levels

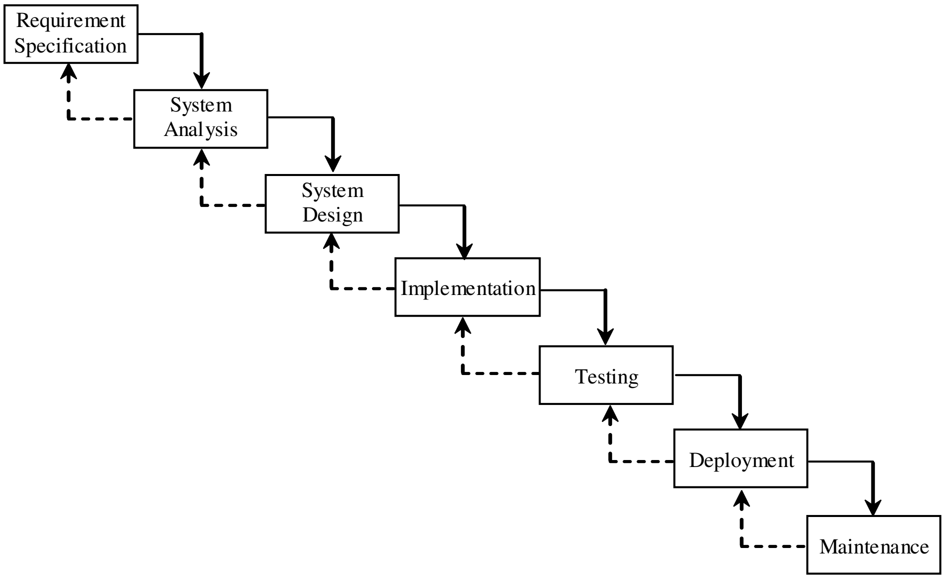
The first level DFD shows the main processes within the system. Each of these processes can be broken into further processes until you reach pseudocode.



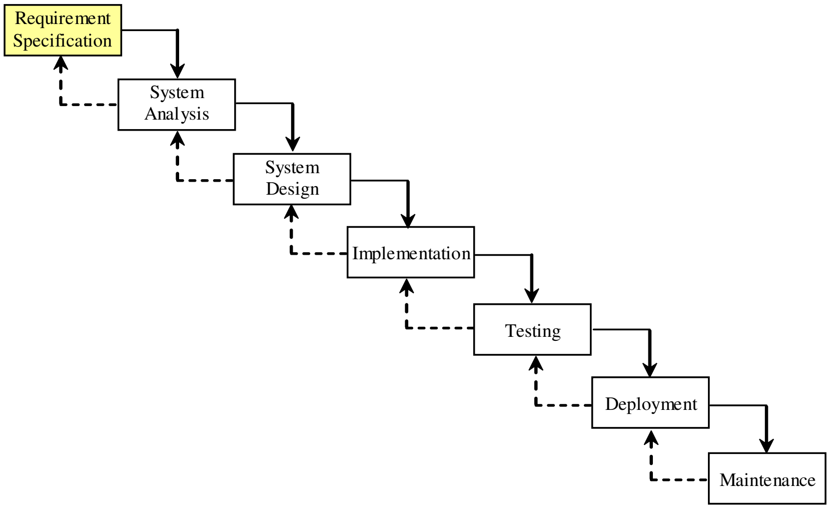
S.NO	RGPV QUESTION	YEAR	MARKS
1.	Write a short note on DFD. Explain with an example of bank account creation and access of the account by a customer or by the bank	Dec,2010	10

UNIT 2/LECTURE 8

object oriented S/W development process model **[RGPV /DEC-2010(10)]**



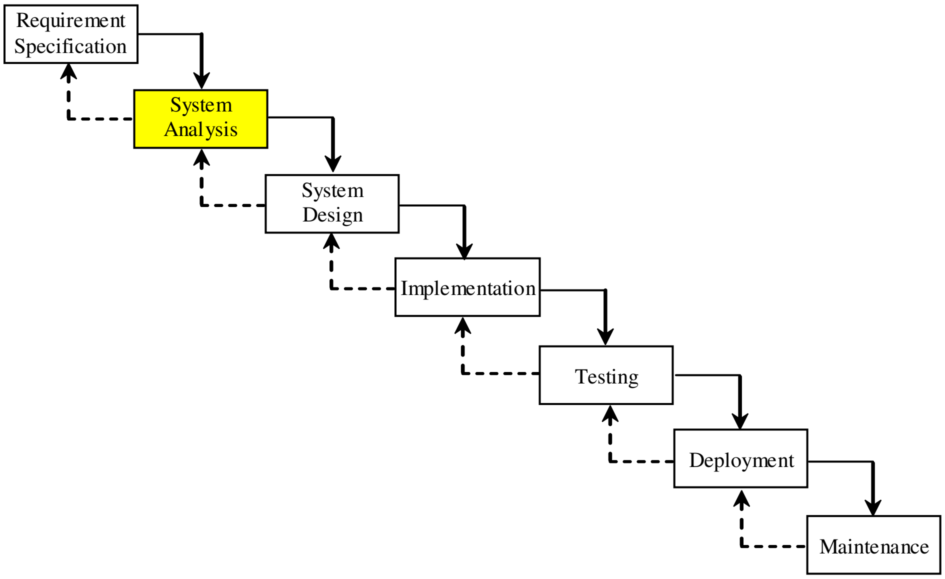
Software Development Process



A formal process that seeks to understand the problem and document in detail what the software system needs to do. This phase involves close interaction between users and designers.

Most of the examples in this book are simple, and their requirements are clearly stated. In the real world, however, problems are not well defined. You need to study a problem carefully to identify its requirements.

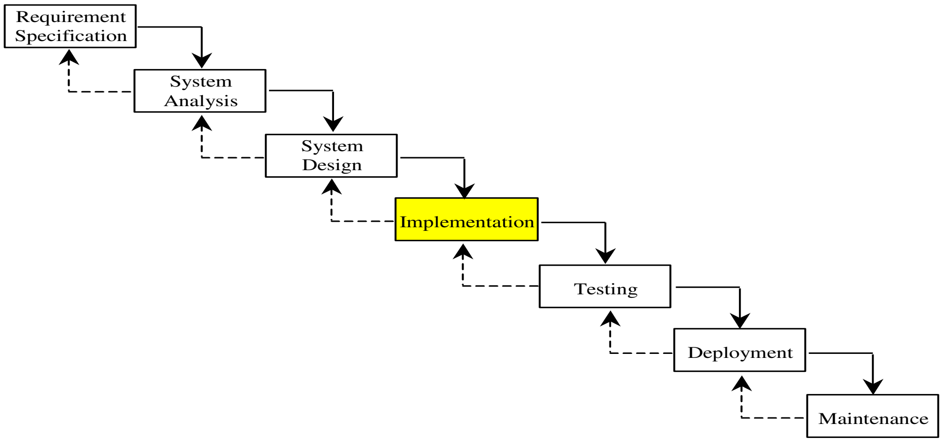
System Analysis



Seeks to analyze the business process in terms of data flow, and to identify the system's input and output.

Part of the analysis entails modeling the system's behavior. The model is intended to capture the essential elements of the system and to define services to the system.

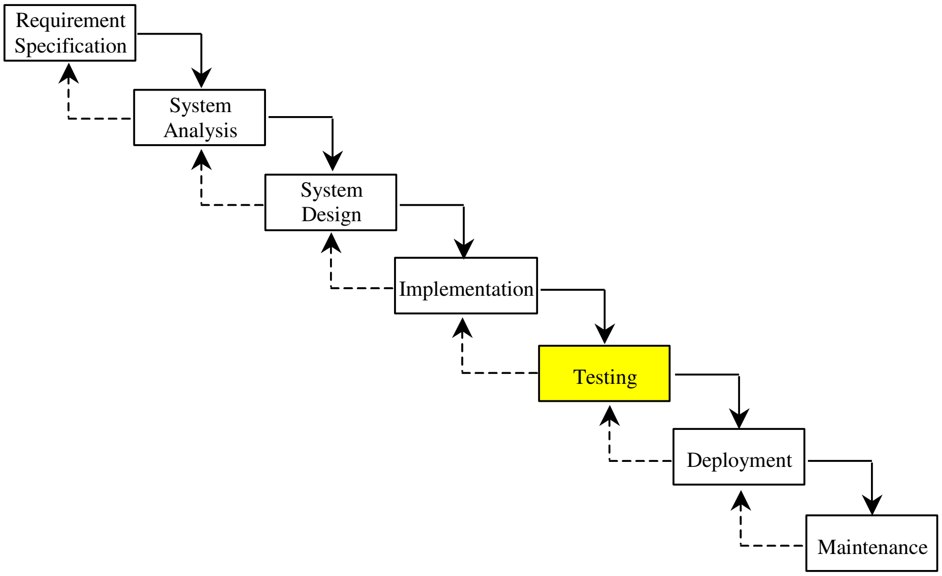
Implementation



The process of translating the system design into programs. Separate programs are written for each component and put to work together.

This phase requires the use of a programming language like Java. The implementation involves coding, testing, and debugging.

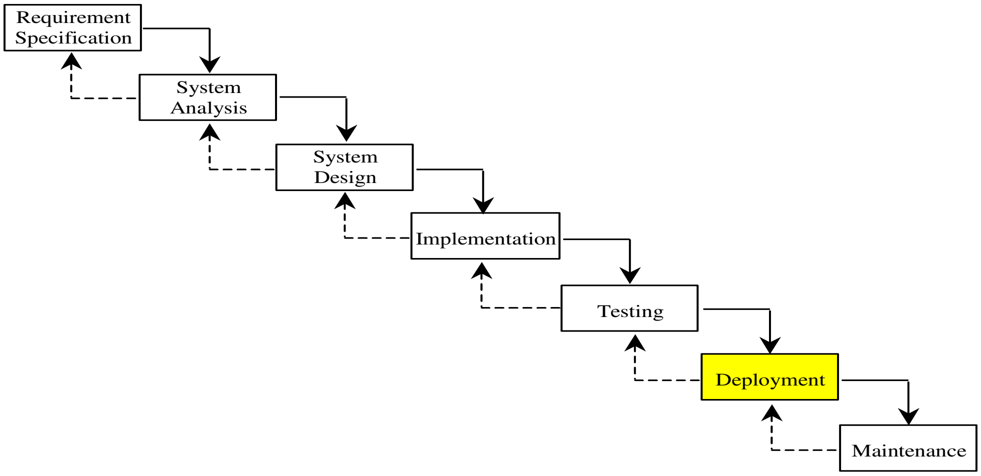
Testing



Ensures that the code meets the requirements specification and weeds out bugs.

An independent team of software engineers not involved in the design and implementation of the project usually conducts such testing.

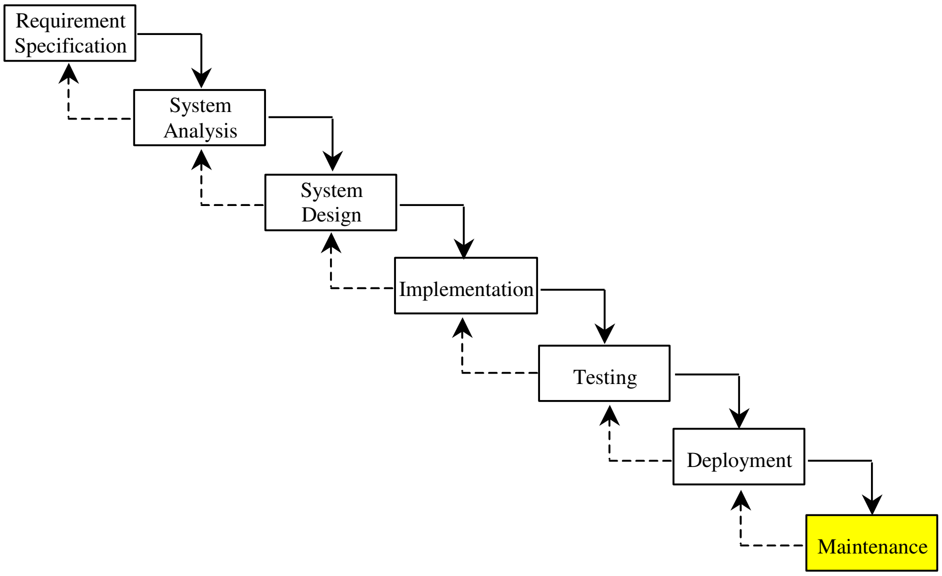
Deployment



Deployment makes the project available for use.

For a Java applet, this means installing it on a Web server; for a Java application, installing it on the client's computer.

Maintenance



Maintenance is concerned with changing and improving the product.

A software product must continue to perform and improve in a changing environment. This requires periodic upgrades of the product to fix newly discovered bugs and incorporate changes

S.NO	RGPV QUESTION	YEAR	MARKS
1.	What is object-orientation in software development and why it is needed?	Dec,2010	10

UNIT 2/LECTURE 9

Analysis **[RGPV /Dec-2009(10),June-2008(5)]**

In academic environments software often seems to grow, without a clear plan or explicit intention of fulfilling some need or purpose, except perhaps as a vehicle for research. In contrast, industrial and business software projects are usually undertaken to meet some explicit goal or to satisfy some need. One of the main problems in such situations, from the point of view of the developers of the software, is to extract the needs from the future users of the system and later to negotiate the solutions proposed by the team. The problem is primarily a problem of *communication*, of bridging the gap between two worlds, the world of domain expertise on the one hand and that of expertise in the craft of software development on the other. In a number of publications object-oriented analysis has been proposed as providing a solution to this problem of communication. object-oriented techniques allow us to capture the system requirements in a model that directly corresponds with a conceptual model of the problem domain

Object-Oriented Analysis

- *analysis = extracting the needs*

Another claim made by proponents of OOP is that an object-oriented approach enables a more seamless transition between the respective phases of the software life-cycle. If this claim is really met, this would mean that changing user requirements could be more easily discussed in terms of the consequences of these changes for the system, and if accepted could in principle be more easily propagated to the successive phases of development.

One of the basic ideas underlying object-oriented analysis is that the abstractions arrived at in developing a conceptual model of the problem domain will remain stable over time. Hence, rather than focusing on specific functional requirements, attention should be given to modeling the problem domain by means of high level abstractions. Due to the stability of these abstractions, the results of analysis are likely candidates for reuse.

The reality to be modeled in analysis is usually very complex. mention a number of principles or mechanisms with which to manage complexity. These show a great similarity to the abstraction mechanisms mentioned earlier.

Analysis methods

The phases of *analysis* and *design* differ primarily in orientation: during analysis the focus is on aspects of the problem domain and the goal is to arrive at a description of that domain to which the user and system requirements can be related. On the other hand, the design phase must result in an architectural model of the system, for which we can demonstrate that it fulfills the user needs and the additional requirements expressed as the result of analysis.

Analysis methods

- Functional Decomposition = Functions + Interfaces
- Data Flow Approach = Data Flow + Bubbles
- Information Modelling = Entities + Attributes + Relationships
- Object-Oriented = Objects + Inheritance + Message passing

Discuss a number of methods that are commonly used in analysis . The choice of a particular method will often depend upon circumstances of a more sociological nature. For instance, the experience of a team with a particular method is often a crucial factor for success.

For this reason, perhaps, an eclectic method combining the various approaches may be preferable (see, for instance, Rumbaugh {\it et al.}, 1991). However, it is doubtful whether such an approach will have the same benefits as a purely object-oriented approach.

The method of *Functional Decomposition* aims at characterizing the steps that must be taken to reach a particular goal. These steps may be represented by functions that may take arguments in order to deal with data that is shared between the successive steps of the computation.

In general, one can say that this method is not very good for data hiding. Another problem is that non-expert users may not be familiar with viewing their problem in terms of computation steps. Also, the method does not result in descriptions that are easily amenable to change.

The method indicated as the *Data Flow Approach* aims at depicting the information flow in a particular domain by means of arrows that represent data and bubbles that represent processes acting on these data.

Information Modelling is a method that has become popular primarily for developing information systems and applications involving databases. As a method, it aims at modelling the application domain in terms of *entities* that may have attributes, and relations between entities.

An *object-oriented* approach to analysis is very similar in nature to the information modelling approach, at least with respect to its aim of developing a conceptual model of the application domain. However, in terms of their means, both methods differ significantly.

The most important distinction between *objects*, in the sense of OOP, and *entities*, as used in information modelling, to my mind lies in the capacity of objects to embody actual behavior, whereas entities are of a more passive nature.

Concluding this brief exploration of the analysis phase, I think we may safely set as the goal for every method of analysis to aim at *stable abstractions* that is a conceptual model which is robust with respect to evolving user requirements. Also, we may state a preference for methods which result in models that have a close correspondence to the concepts and notions used by the experts operating in the application domain.

S.NO	RGPV QUESTION	YEAR	MARKS
1.	Differentiate between analysis and design.	June,2008	5
2.	How will you define the object oriented analysis approach?	Dec,2009	10