Notes On

Object Oriented Software Engineering

Central Department of MscCSIT,
Tu, Kirtipur

Table of Contents

Unit 1

1.0 Questions

i. Explain Object Oriented **Software development** with Example. [10 Marks, 2071]

ii. Explain **object orientation development process** and **object-oriented analysis** with example. [10 marks, 2072]

iii. What is requirement model ? Explain with practical example. [2072]

iv. Differentiate between model architecture and requirement model. [2071]

v. Explain with example software process model and a software process. [10 Marks, 2075]

vi. Differentiate between **object-oriented software engineering** and **object-oriented software development**.[10 Marks, 2074]

[Hint: Software development is done by using different process model like waterfall, spiral]

vii. Explain about **object-oriented system design** and **object-oriented program design**. [10 Marks, 2074]

SN	**Software development life cycle**	**System development life cycle**
1	It only looks at software components development planning, technical architecture, software quality testing and deployment of working software	It involves end-to-end People, process, Software/Technology deployment. This includes Change Management, training, Organizational updates, also.
2	is about building a software ("only") in a phased approach systematically.	System development life cycle is about implementing hardware and software in a phased manner systematically.

Object-oriented software engineering

- Object-Oriented Software Engineering (OOSE) is a software design technique that is used in software design in object-oriented programming.
- It is also called as objectory is a method of object–oriented development with the specific aim to fit the development of large, real – time systems
- OOSE is developed by **Ivar Jacobson** in 1992.

- The development process called use–case driven development stresses that use cases are involved in several phases of the development including analysis, design, validation and testing.
- OOSE is the first object-oriented design methodology that employs use cases in software design.
- The use–case scenario begins with a user of the system initiating a sequence of interrelated events.
- Objectory has been developed and applied to numerous application areas and embodied in the CASE tool systems.
- Objectory is built around several different models:
  i. Use–case model: It defines outside (actors) and inside (use case) of the system’s behavior

  ii. Domain object model: The objects of the real world are mapped into the domain object model

  iii. Analysis object model: It presents how the source code (implementation) should be carried out

  iv. Implementation model: It represents the implementation of the system

  v. Test model: It constitutes the test plans, specifications and reports
- OOSE is one of the precursors of the Unified Modeling Language (UML), such as Booch and OMT.

- It includes a requirements, an analysis, a design, an implementation and a testing model.
- Interaction diagrams are similar to UML's sequence diagrams. State transition diagrams are like UML state-chart diagrams.

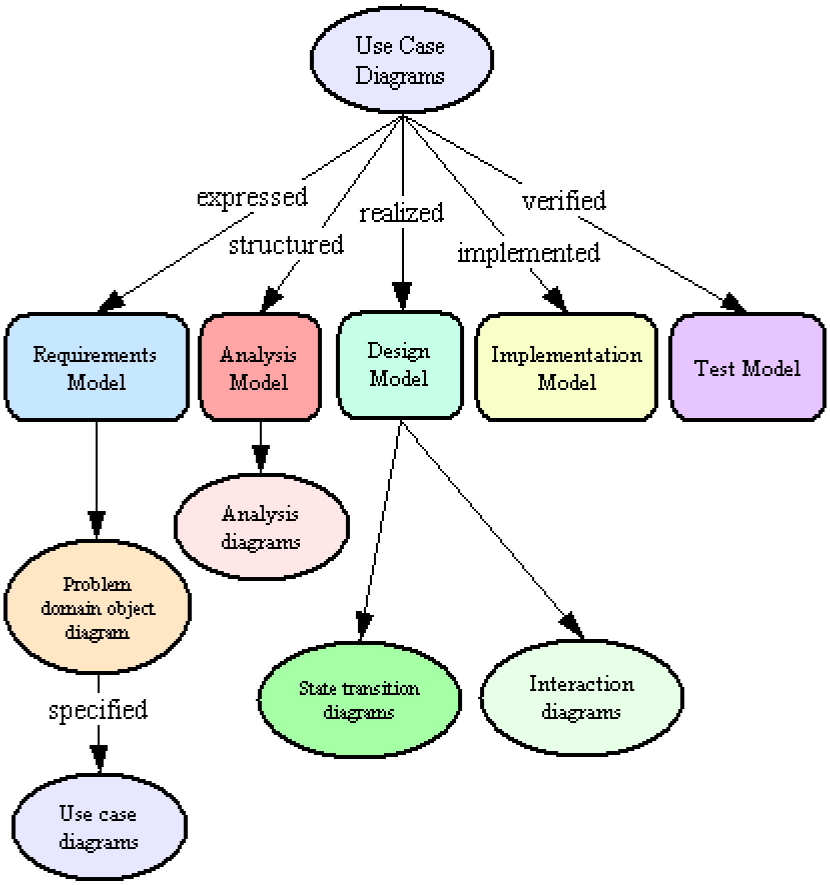


Fig: object-oriented software engineering

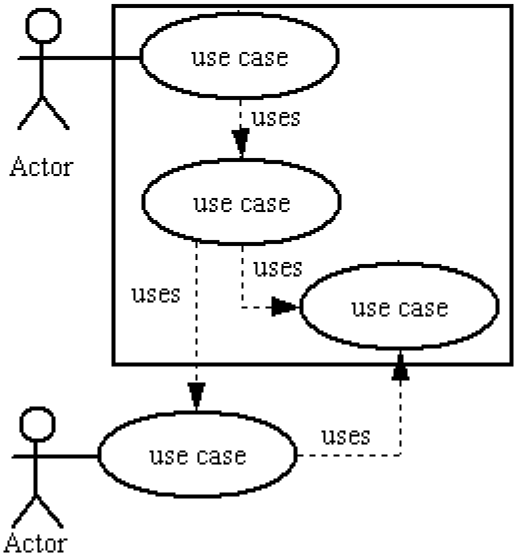


Fig: use case diagram

Object-oriented Analysis

- This phase of s/w development is concerned with determining the system requirements and identifying classes and their relationship to other classes in the problem domain.
- Scenarios are a great way of examining who does what in the interactions among objects and what role they play; that is, their interrelationships. This intersection among objects' roles to achieve a
- given goal is called collaboration.
- In essence, a use case is a typical interaction between a user and system that captures users' goals and needs. Expressing high level processes & interactions with customers in scenario & analyzing

it is referred to as use–case modeling. It represents users' view of the system or users' needs.

- Looking at the physical objects in the system also provides us important information on objects in the systems. The objects could be individuals, organizations, machines, units of information,
- pictures, or whatever else makes sense in the context of the real–world systems.
- In regarding documentation, 80 – 20 rule is generally applies where 80 percent of the work can be done with 20 percent of the documentation. Good modeling implies good documentation.

Object-Oriented Analysis Unified Approach:

- The goal is first to find domain of the problem and system's responsibilities by knowing all user needs which is accomplished by models.
- These models concentrate on describing what the system does rather than how does it. OOA has the following steps:

  i. Identify the Actors,

  ii. Develop a simple business process model using UML Activity diagram,

  iii. Develop the Use Case,

  iv. Develop interaction diagrams,

  v. Identify classes

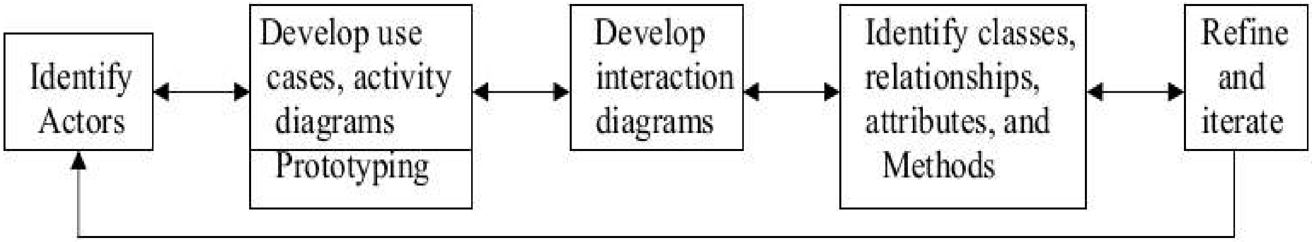

Identify Actors
Develop use cases, activity diagrams
Prototyping
Develop interaction diagrams
Identify classes, relationships, attributes, and Methods
Refine and iterate

Fig: O-O-Analysis

Object-oriented Design (Object Oriented program design):

- The goal of OOD is to design the classes identified during the analysis phase and the user interface.
- Here, we identify & define additional objects, classes that support implementation of requirements.
- OOD is highly incremental. OOA can done, model it, create OOD them do some more on it.
- Here, we first, build the object model based on objects and their relationships, then iterate and refine the model such as
  - i. Design and refine classes
  - ii. Design and refine attributes
  - iii. Design and refine methods
  - iv. Design and refine structures
  - v. Design and refine associations.
- Guidelines to use in OOD:
  - i. Reuse, rather than build, a new class. Know the existing classes
  - ii. Design a large number of simple classes, rather than a small number of complex classes

iii. Design methods

iv. Critique what you have proposed. If possible, go back and refine the classes.

Object-Oriented Design: Unified Approach:

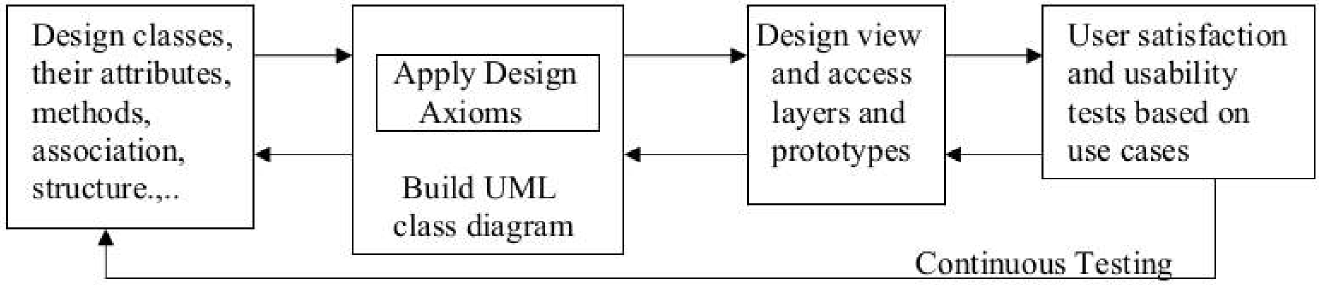

Design classes, their attributes, methods, association, structure.,..
Apply Design Axioms
Build UML class diagram
Design view and access layers and prototypes
User satisfaction and usability tests based on use cases
Continuous Testing

Fig: O-O-Design

UA combines Jacobson's analysis and interaction diagrams,

Booch's object diagrams & Rumbaugh's domain models to get good design. OOD step consists of

i. Designing classes, attributes, methods, associations, structures & protocols, apply design axioms

ii. Design the Access Layer & Design and prototype User interface

iii. User satisfaction and Usability Tests based on Usage / Use cases

iv. Iterate and refine the design

The Software Process:

- Coherent sets of activities for specifying, designing, implementing and testing software systems
- A structured set of activities required to develop a software system

i. Specification

ii. Design

iii. Validation

iv. Evolution

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective

Software Life Cycle Models/ Software process models:

Waterfall Model:

- first SDLC Model to be used widely in Software Engineering.
- In "The Waterfall" approach, the whole process of software development is divided into separate phases.
- The outcome of one phase acts as the input for the next phase sequentially.

Best situation to use this model:

- requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and not dynamic.
- There is no ambiguous requirement.
- The projects is short.

Cons of this model:

- hard to adopt to software requirement changes.
- Difficult to estimate time and cost for the phases.
- Handling risk is not part of this model.

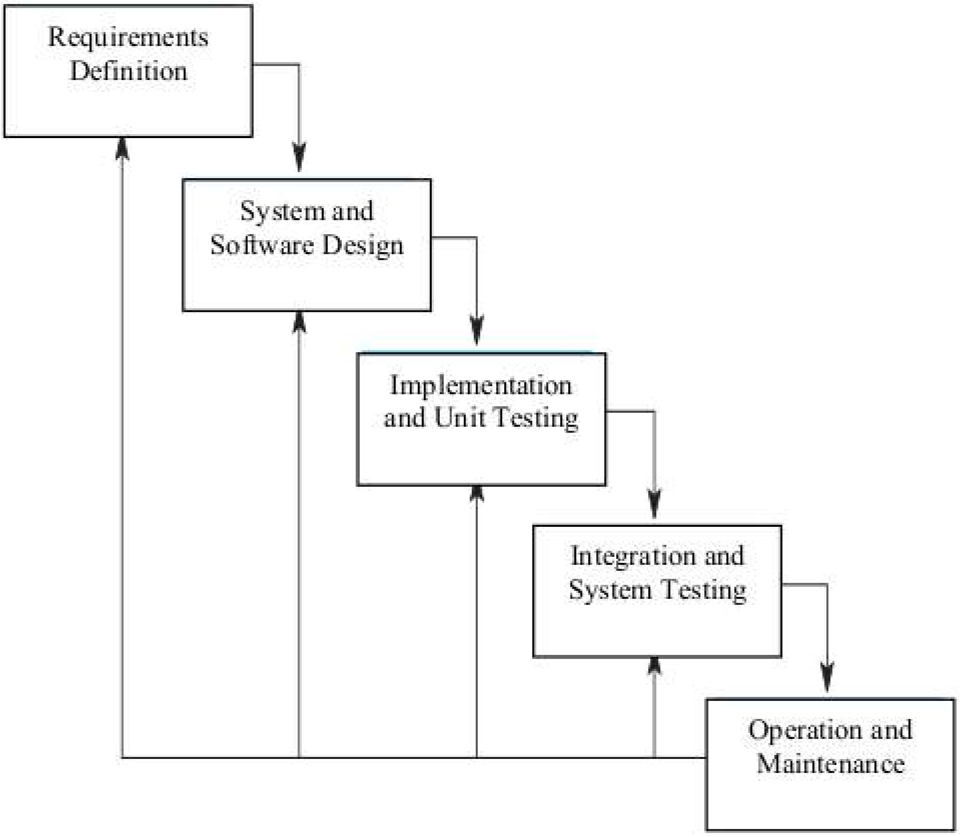

Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

Fig: Waterfall Model

V-shaped model:

- it's a variation of waterfall model which more emphasized on testing.
- It's a sequential model each phase must be completed before next phase begins.
- It provides simple and easy to follow map of the software development process.

Best situation to v-model:

- v-model can be used for small to medium size projects where requirements are clearly defined and fixed.
- It can choose when ample technical resources are available and with needed technical expertise.

Cons:

- It is too simple to accurately reflect the software development process.
- It is inflexible; it has no ability to respond changes.
- It produces inefficient testing methodologies.

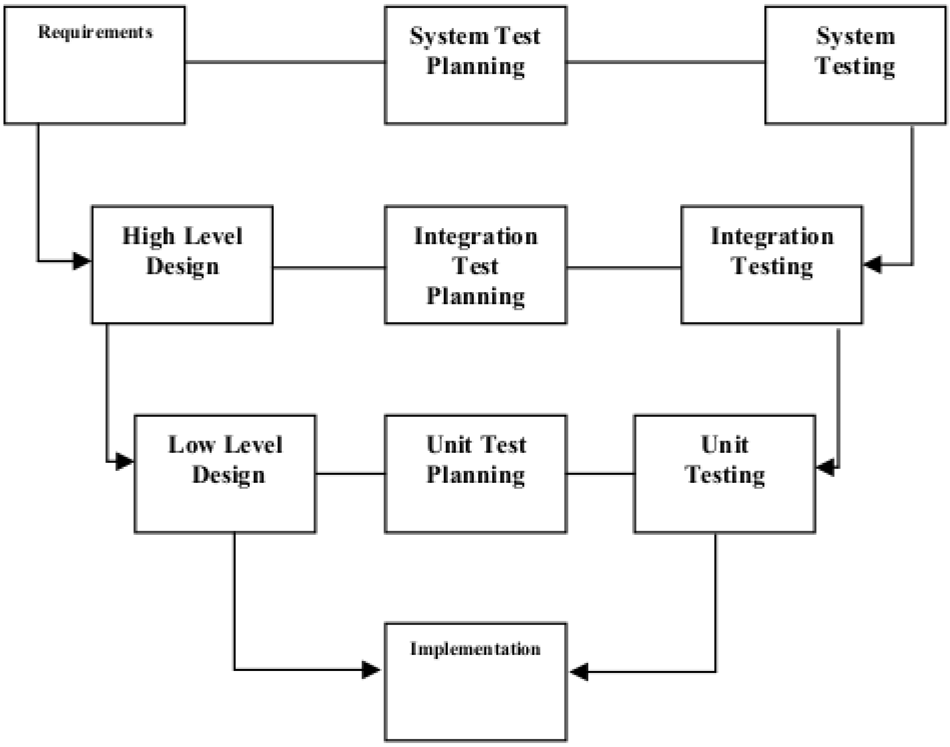


Fig: V-model

Spiral Model:

- The spiral model is similar to the incremental model, with more emphases placed on risk analysis.
- The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.
- A prototype produced at the end of risk analysis phase.
- A software project repeatedly passes through these phases in iterations (called Spirals in this model).

Advantages:

- High amount of risk analysis, hence avoidance of risk is enhanced.
- Spiral can be used for large and mission critical projects.
- Additional functionality can be added at a later date.
- Software is produced early in the software life cycle.

Disadvantage:

- can be costly
- Risk analysis requires highly specific expertise.
- Projects success is highly dependent on the risk analysis phase.
- don’t work well for small projects

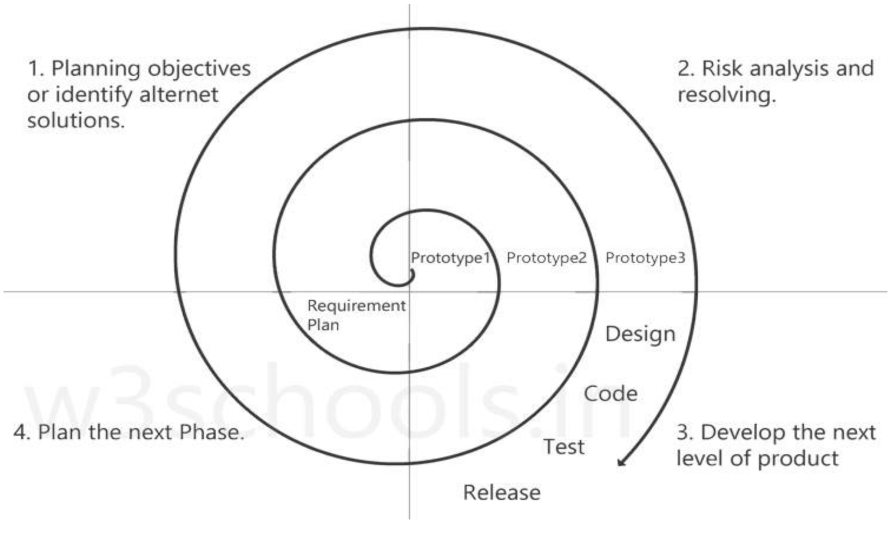

1. Planning objectives or identify alternet solutions.
2. Risk analysis and resolving.
Prototype1
Prototype2
Prototype3
Requirement Plan
Design
Code
Test
Release
4. Plan the next Phase.
3. Develop the next level of product
w3schools.in

Fig: Spiral Model

Comparison of various SDLC Model

Features/ Models	Waterfall model	Iteration model	V-shaped Model	Spiral Model	Extreme model
Understanding Requirements/ Requirements Specification	Well understood at Beginning	Well understood at Beginning	Well understood at Beginning	Not Well understood at Beginning	Requirements are recorded on story card.
Duration	Long	Not very Long	Long	Long	Short -medium
Cost	Low	Medium	Medium	Expensive	Expensive
Documentation and Training required	Vital	Yes	Vital	Yes	As per requirement
Guarantee of Success	Less	High	Higher than Waterfall	High	High
Risk	High	Medium	Medium	Low	Low
User Involvement	Low	High	Low	High	Full-time customer engagement
Flexibility	Rigid	Less Flexible	Little Flexible	Flexible	Flexible
Simplicity	Simple	Intermediate	Simple	Intermediate	Simple
Implementation	Easy	Easy	Easy	Complex	Easy
Overlapping Phases	No	No	No	Yes	Yes
Use	Small project	Long running projects	Small projects	Complex large Projects	small-medium size projects

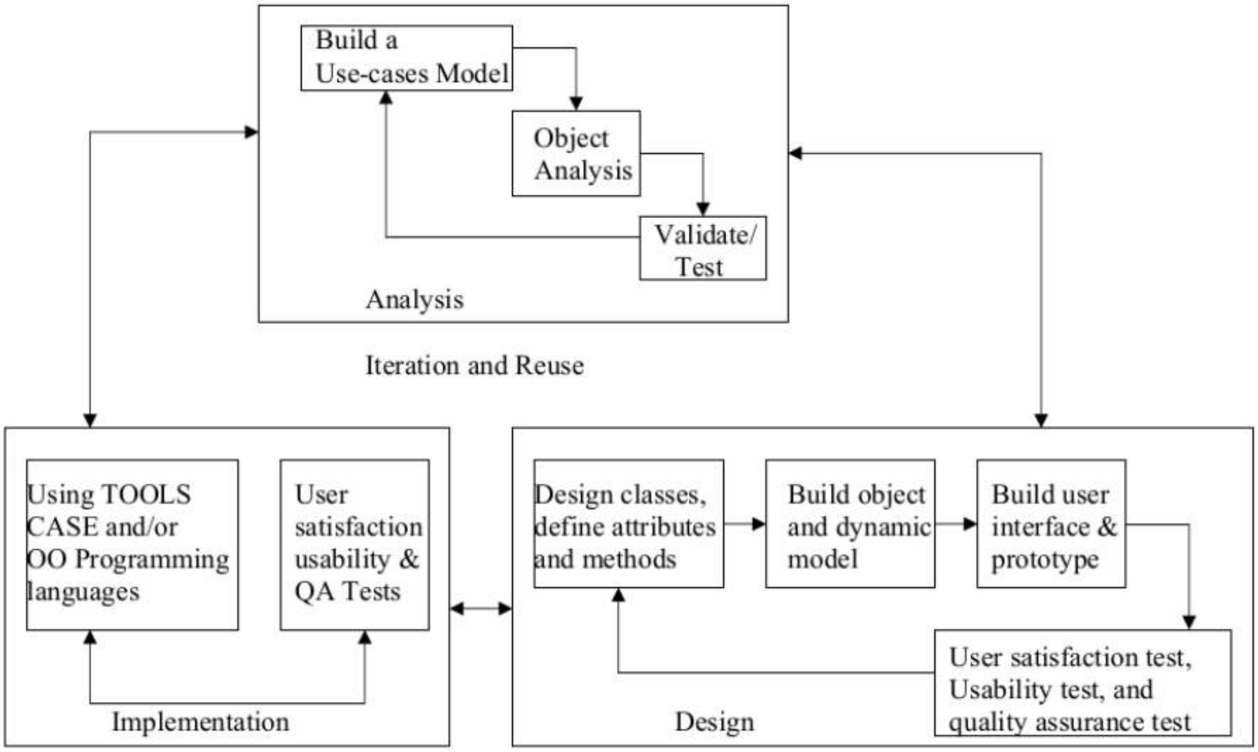
Object Oriented Software development

- The object oriented software development life cycle (SDLC) consists of three macro processes:

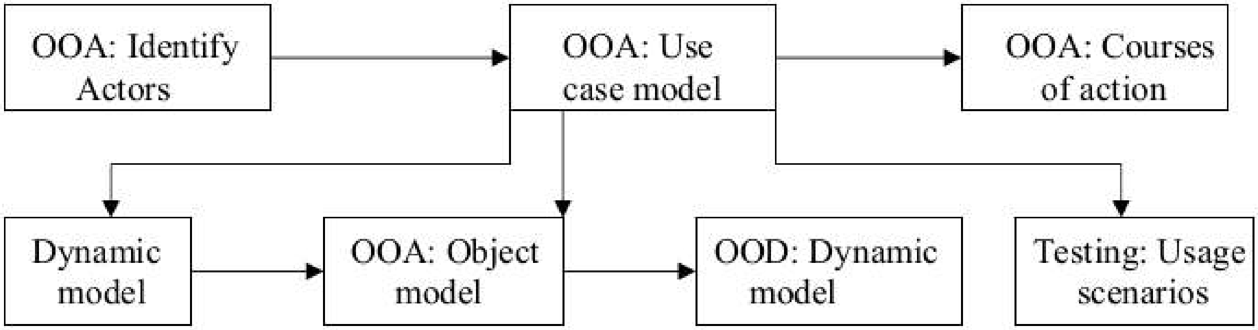
  i. object-oriented analysis

  ii. object-oriented design

  iii. object-oriented implementation.



- Produce designs are traceable across requirements, analysis, design, implementation and testing and it can be traced back directly to user requirements.



- Object–oriented system development includes object–oriented analysis, object–oriented design, Prototyping, Component–based development, Incremental testing.

Object-oriented System development

→ System Development

- SD refers to all activities that go into producing an information systems solution. These activities consist of requirement analysis, modeling, design, implementation, testing and maintenance.
- A software development methodology is a series of processes that if followed can lead to the development of an application.
- According to Niklaus Wirth – A software system is a set of mechanisms for performing certain action on certain data.
- There are two orthogonal views of the software differs in their primary focus.

- ○ The traditional approach focuses on the functions of the system and says software as a collection of programs (or functions) and isolated data.

- ○ Object oriented systems development centers on the object, which combines data and functionality i.e., Programs = Algorithms + Data Structures.

→ **Object Oriented System Development**

- Object oriented systems development is a way to develop software by building self – contained modules or objects that can be easily replaced, modified and reused.

- In an object–oriented environment, software is a collection of discrete objects that encapsulate their data as well as the functionality of real–world events "objects" and emphasizes its cooperative philosophy by allocating tasks among the objects of the applications.

- Software Oriented system development can be divided into five phases.
    - ○ Analysis
    - ○ design
    - ○ implementation

- testing
- refinement

- The development is a process of change, refinement, transformation or addition to the existing product. The software development process can be viewed as a series of transformations, where the output of one transformation becomes input of the subsequent transformation.

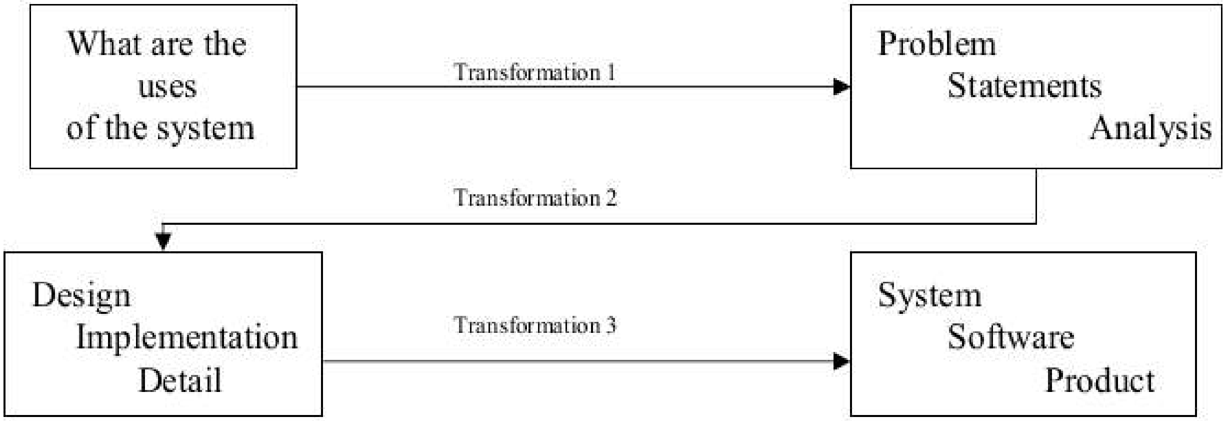

What are the uses of the system
Transformation 1
Problem Statements Analysis
Transformation 2
Design Implementation Detail
Transformation 3
System Software Product

Fig: system development process

- Transformation 1 (analysis) translates the users' needs into system requirements & responsibilities.
- Transformation 2 (design) begins with a problem statement and ends with a detailed design that can be transformed into an operational system. It includes the bulk of s/w development activity.
- Transformation 3 (implementation) refines the detailed design into the system deployment that will satisfy the users' needs. It represents embedding s/w product within its operational environment.

- An example of s/w development process is the waterfall approach which can be stated as below.

Unit 2

Questions

i. Explain function/data methods in object oriented system development with Example. [5 marks, 2072]

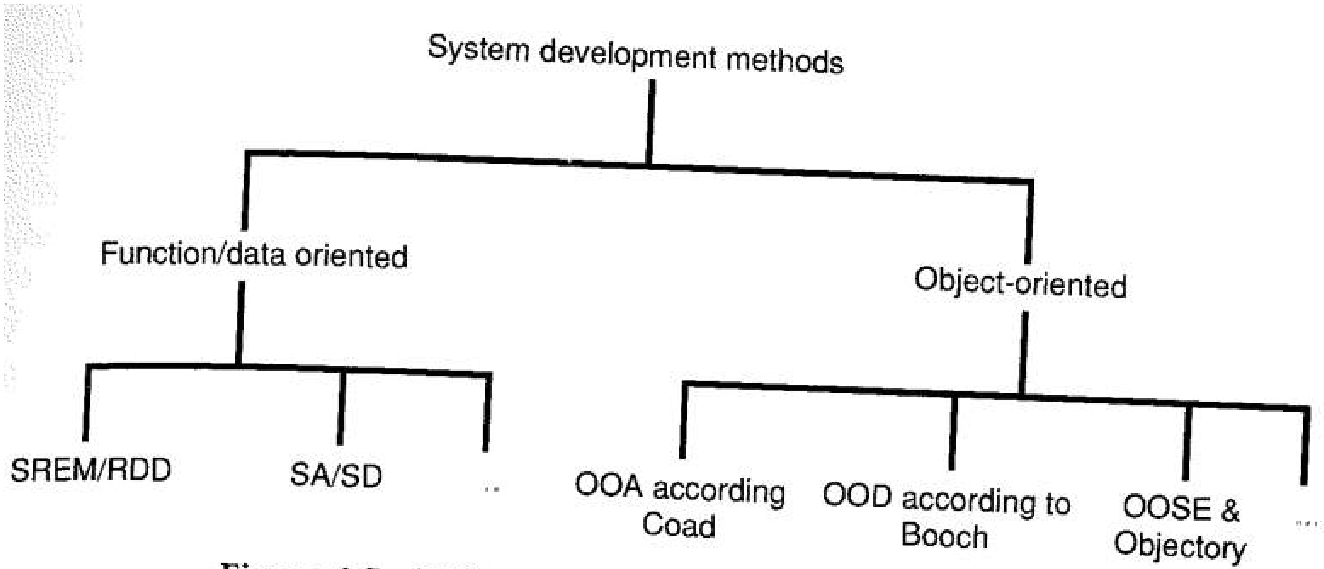


Figure 4.5 Different software development paradigms

Function/Data Methods:	**object Oriented method**
• In function data methods, function is active and have behavior, and data is passive information holder which is affected by the function.	• In Object Oriented method, function and data is combined as integrated objects.
• The system is typically broken down into functions whereas	• The system is broken down into objects, function and data will

data is sent between those functions.	be method and properties of that object.
• A system developed using function/data method often becomes difficult to maintain.	• This method is easier to understand and thus easier to maintain than the result of function/data method.
• In function/data method approach all function must know how the data must be stored, i.e data structure.	• Each object should be defined internally, which includes defining the information that each object must hold.
• System generated using this methods often quite unstable; a slight modification will generate major consequences.	• Items with low modification probability are naturally identified and it is possible to isolate at an early stage items with a high probability of modification.
• The requirement specification is normally formulated in normal human language not in function/data structure, which creates large semantic gap between the external and internal views of the system.	• Object is a naturally occurring entities in the application domain. So this method will model the application domain more efficiently and reduces the semantic gap between them.

Unit 3

3.0 Questions

i. What is requirement model ? Explain with practical example. [2072]

ii. Explain with example of different steps to develop the requirement model from the user requirements. [10 Marks, 2074]

iii. Differentiate between model architecture and requirement model. [2071]

iv. Comparison between components and component management. [2075]

v. What is a component ? Explain component management with example. [2074]

vi. What are the main reasons in construction phase ? What is done in construction phase ? Explain. [10 Marks, 2071]

vii. Differentiate between model architecture and requirement model. [10 Marks, 2071]

viii. Explain the classification of the real time system with example. [5 Marks, 2072].

ix. Requirement Engineering

3.1 Architecture, Analysis, Construction,RTS, DBMS, Component, Testing

Model Architecture:

- System development is basically concerned with developing models of the system i.e. identifying and describing objects in certain information space.

- There are several ideas regarding to find a good object, however good object doesn't exist on its own.
- An object can be perfectly right for one model, but totally wrong in another model.
- The important criteria is that it should robust against modification and should help to understand system. As we know for the certain that the all systems we build will be modified, so we must create a robust model structure. Therefore we must analyze how modification will affect the system
- after we have worked with a model for a while, a stable structure will evolve for the system. By working a long time with the early models, we will obtain a good understanding of the system.

Requirement Models:

- Requirement models defines the system boundaries and functionality that should offer .
- It function as contact between developer and the ordered of the system, specially from developers view of what customer wants.
- This model should be readable also for non OOSE practitioners.
- This model govern the development of all other model so this model is central one throughout the whole system.

- The requirement model will be structured by analysis model, realized by design model, implemented by implementation model and tested in testing model.
- The requirements model consists of **three parts:**

  i. Use case model

  ii. problem domain object model and

  iii. interface.

Use case model:

- A use case is a specific way of using the system by performing some part of functionalities.
- A use case constitutes complete course of event initiate by the actor and it specifies the interaction between the actor and the system.
- The collected use case specify all the existing way of using the system.

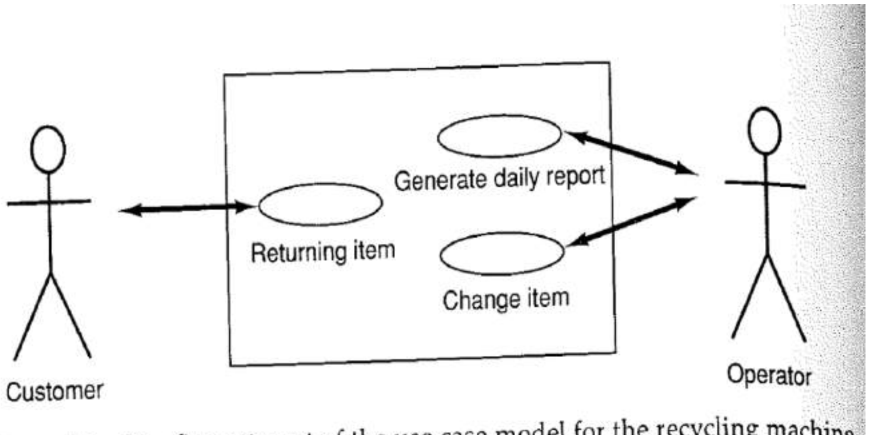


Figure 7.4 The first attempt of the use case model for the recycling machine.

Interface Description:

- We should define the interface in details while we describes the the use cases and communicating to potential users.
- We can simulate the use cases with sketches of what user will see in the screen and for sophisticated simulation use UIMS(user interface management system)
- This model guarantee that the user interface will be consistent with the user's logical system perspective.
- In the recycling machine the user interface is quite trivial (being mainly a push-button machine)

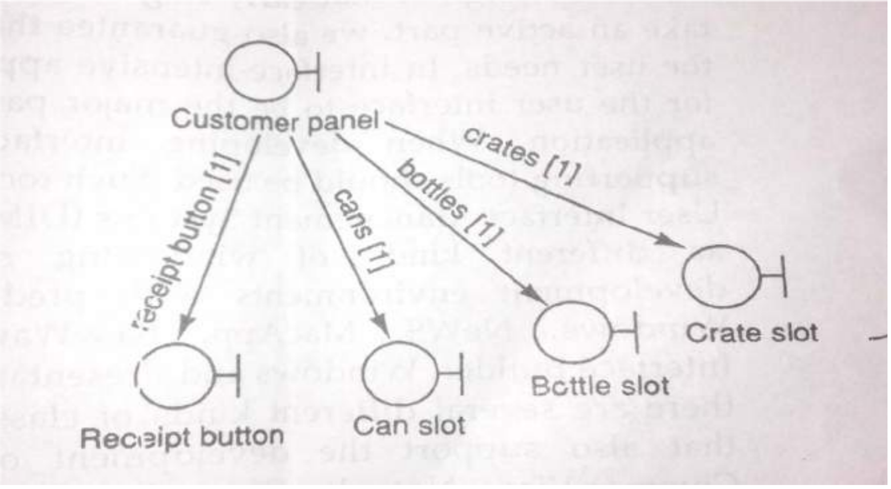


Fig: Example of interface design

Problem Domain Object:

- When requirements specification exists in very vague form, it is difficult to define task and delimitation(boundary) of a system.
- So, it is good to develop a logical view of the system using problem domain objects.

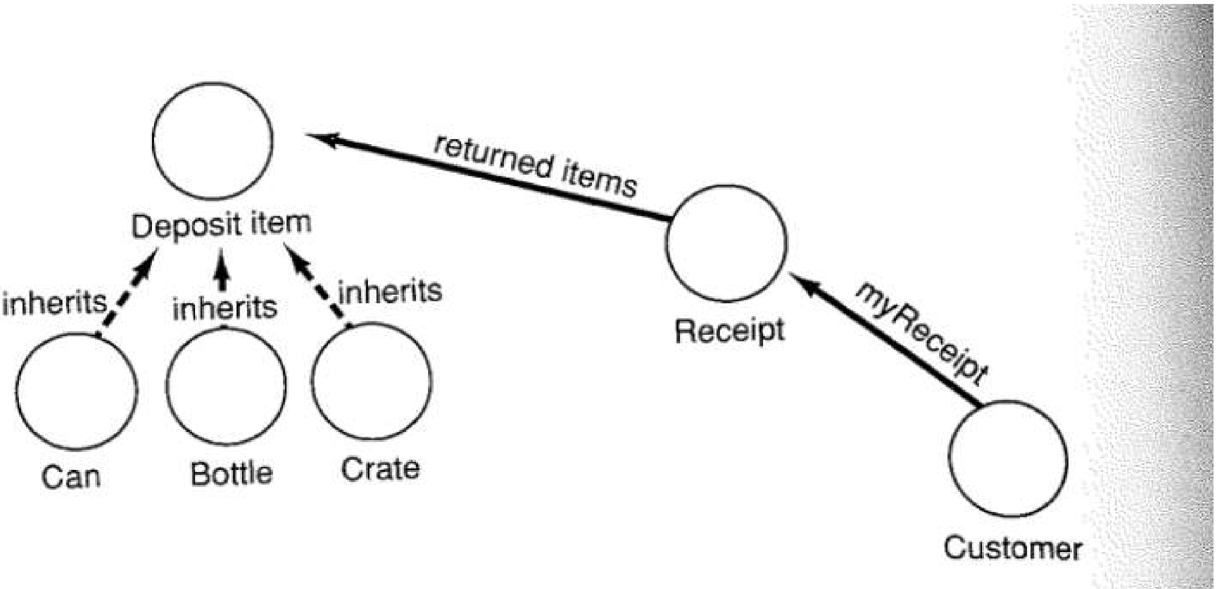


Figure 7.8 A problem domain model of the recycling machine.

Q.1 what are the main reasons in construction phase ? What is done in construction phase ? Explain.

There are three main reasons for having construction phase.

- **Analysis model is not sufficiently formal:**

To change the source code we must refine the object; which operation should we offer, exactly what should the communication between different objects look like etc.

- **the actual system must be adopted to implementation environment:** In analysis we assumed an ideal world for our system. Actually there is no ideal world and we must adopt to the environment in which system the system is to be implemented.
- **To validate the analysis results:** In construction we see the results of analysis, if we discover unclear

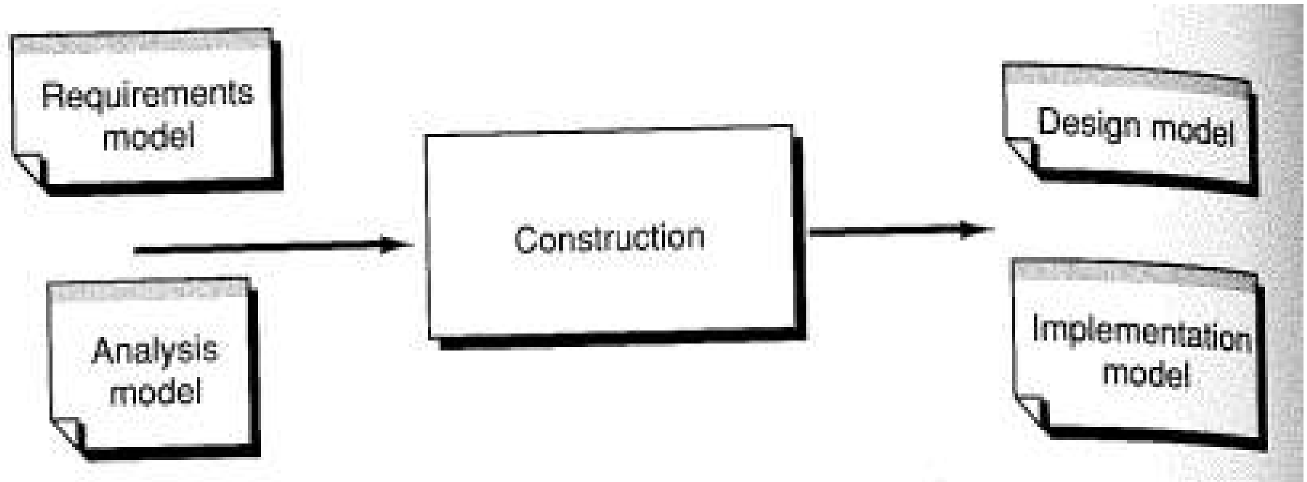


Figure 8.2 The input and output models of construction.

d **implementation model**, thus construction model is further divided into two phases; design and implementation. The design model is a further refinement and formalization of the analysis model where consequences of implementation environment have been taken into account. The implementation model is actual implementation(code) of the system.

[Todo: Read more on text book page **219**]

Q.1 Explain the classification of the real time system with example. [5 Marks, 2072].

Real Time system:

- A real-time system is a system where the correctness depends not only on the logical result of computation but also on the time at which the results are produced
- Ex: Control of modern aircraft, telephone exchange

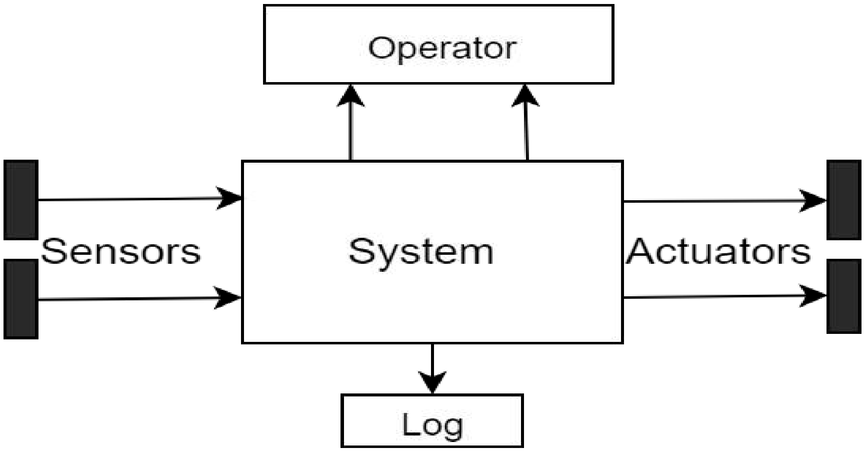


Fig: Abstract Model of RTS

- Figure shows **Sensors** and **Actuators** providing real time view of application Behaviors.
- RTS has means of observing what's going on, Controlling processing from operator and keeping history via logging.

Classification of real-time system:

It can be classified into:

i. Hard real-time system

ii. Soft real-time system

SN	Hard Real Time System	Soft Real Time System
1.	System with hard deadline that must meet otherwise a catastrophe can occur.	System where services are provided in real time but catastrophe will not occur if immediate service is not provided.
2.	Deterministic predictability of processes execution is essential property.	The execution of resources are stochastically distributed based upon the quantities of resources available and the loading of the system.
3.	Ex: control of modern air craft	Digital telephone exchange

Component:

- A component is a standard building unit in an organization that is used to develop application.
- Ex. For a Car manufacture; tires, steering wheel, gear box, doors, seats, engine, gas tank can be the components which may or may not be further divided to smaller components.

- To make a component reusable in various applications, it must be be independent of the application for which it was designed. (but not, necessarily, always)
- component must be designed to be reused.
- It must includes all the reasonable operations **that makes it meaningful to use for any other developers** but not other operations which makes component hard to understood and use.
- It should provide general abstraction so that it can be widely used.
- Components need to be well tested and well documented so that it gets accepted by developers. But this makes component expensive to develop.
- Components that one can modify according to one's need or component that needs to be modified before reusing it is called **white box component.** Ex. Web Framework like Laravel/Symfony for PHP, Spring for Java,
- Component that doesn't need to modify for use or components which are designed to solve specific domain problem are called **black box component**. Ex. Moment js (Date manipulation plugin)

Use Of Components: Component could be used in following places:

- **General Entity Object:**

an entity object that are used to develop other entity object and whose information should be stored in a database, a general framework can be developed as a component. Ex. ORM(object relational Mapping)

- **Interface Object:**

  Interface objects can be implemented using components. Ex. For windows systems, windows buttons and scroll bars are typical components. Similarly one application could have similar interface which then could be a general framework.

- **Some Control Objects:**

  General function such as logging activities, data collection, for statistics, online-help etc possibly used in multiple places and even in multiple application which could be a reusable framework.

- **Different kinds of types:**

  different kinds of types will occur at various phases for example attributes types, types of parameters, and local types, when implementing the blocks some of this types are general and can be implemented as component.

Component Management:

The development of component is often more expensive that the development of ordinary software that's why the component system is normally not profitable for only one project. The real benefits come when a component could be use on multiple projects. T**hus component management should be based on multiple projects.**

A special component management department or group is therefore necessary, that builds component library for the organization and also obtains components from the market. If such department doesn't exist, a spontaneous component activity will evolve on the individual, group or project level, which should be encouraged by the project management since it could form a source of good component.

There are two types of activities for component management.

- One for the design of complete component system.
- One for design of individual components.

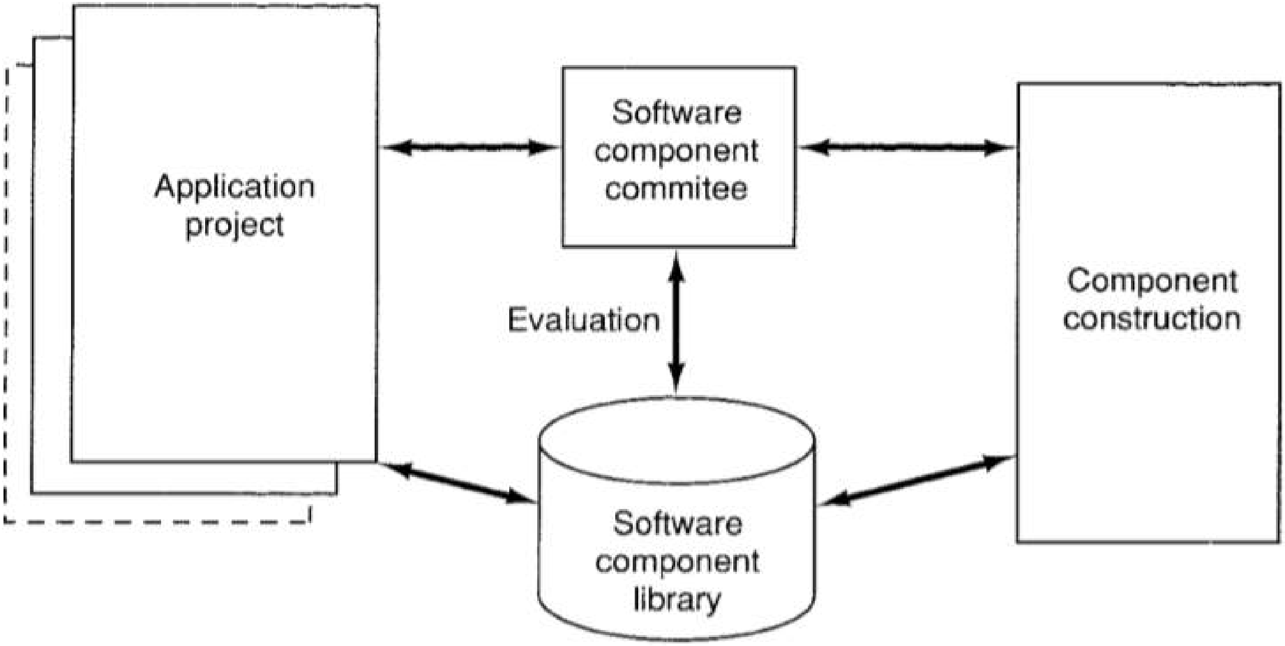


fig: An organization of component management

Unit 4

4.0 Question

i. Explain Software Matrics with Example. [5 Marks, 2073]

ii. Explain project selection and preparation of object oriented software development. [5 Marks, 2073]

iii. Explain Key factors involve in managing the object-oriented software Engineering. [5 Marks, 2072]

iv. What do you mean by software quality assurance ? Explain. [5 Marks, 2072]

4.1 Managing Object-oriented Software Engineering

Q. Explain Key factors involve in managing the object-oriented software Engineering.

→ Key factor involved in managing object-oriented software engineering are:

i. Project selection and preparation
ii. product development organization
iii. project organization and management
iv. Project staffing
v. Software Quality assurance
vi. Software Metrics

Project selection and preparation

- Select a real project that is important, but not with a tight time schedule or any other hard constraint.
- Select a problem domain that is well known and well defined.
- Select people experienced in system development who have positive view of changes. The management should have confidence in them.
- Select a project manager with high degree of interest in the task.
- The staff should work full time in the projects and not be distracted by other projects.

- Base your work on a detail plain developed in advance. Perform evaluation at all stages with criteria established in advance.

Preparation:

- All personnel involved in the new order of work need education and training.
- Give strict method process definition, more emphasis can be put on formal education and training.
- A new development process involves a lots of changes which brings potential risks.
- These risk can be managed by three simple steps:

    i. Risk identification

    ii. Risk valuation

    iii. Managing the risks

Product development Organization

- Product development is to develop different models in sequence.
- The first model to be development is requirement model.

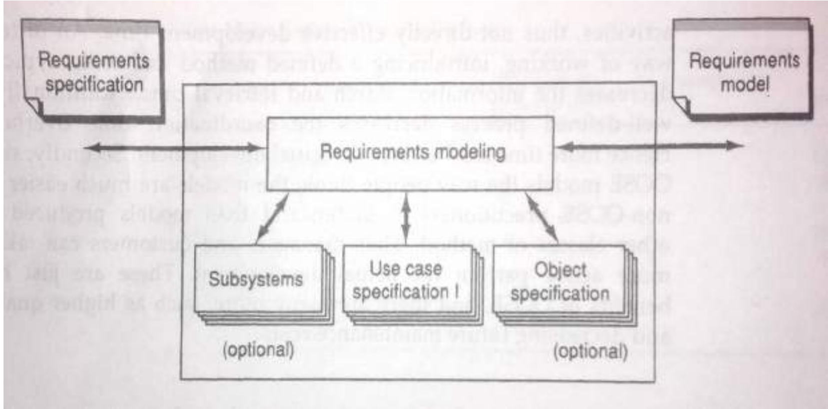


Fig: The process of requirements analysis

- The requirements analysis process delivers a well-defined result: the requirement model with the use case specification.

- The next is analysis process which forms the well defined result process model.

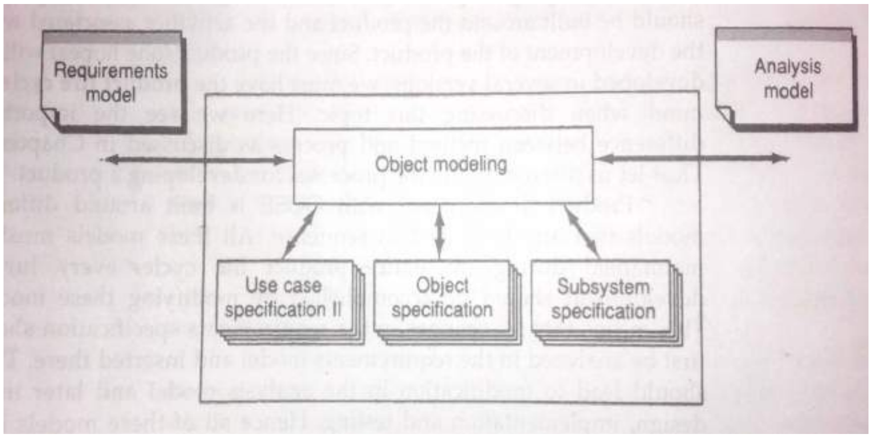


Fig: The process of robustness analysis

- Analysis model is the input for the construction process. The construction process has three sub-process. They are use case design, block construction and subsystem construction.

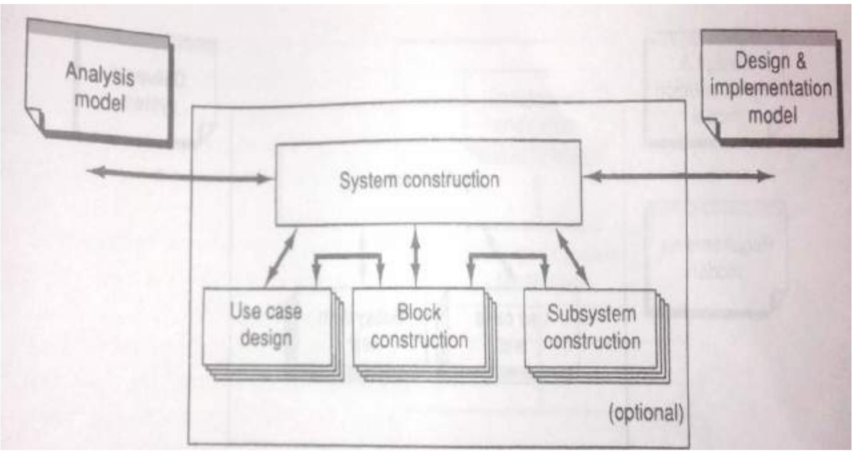


Fig: The process of construction.

- The well defined result delivered by construction process is design model and source code for the unit-test blocks.

- The next is testing process

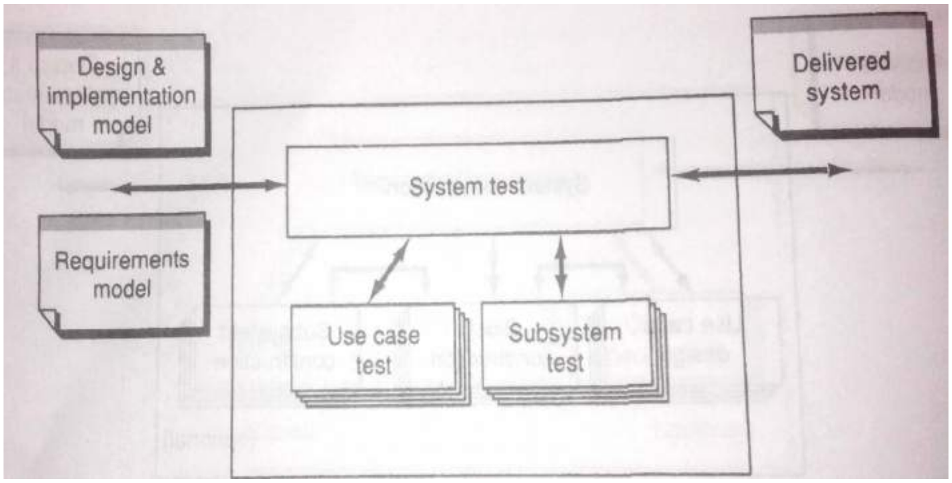


Fig: Testing Process

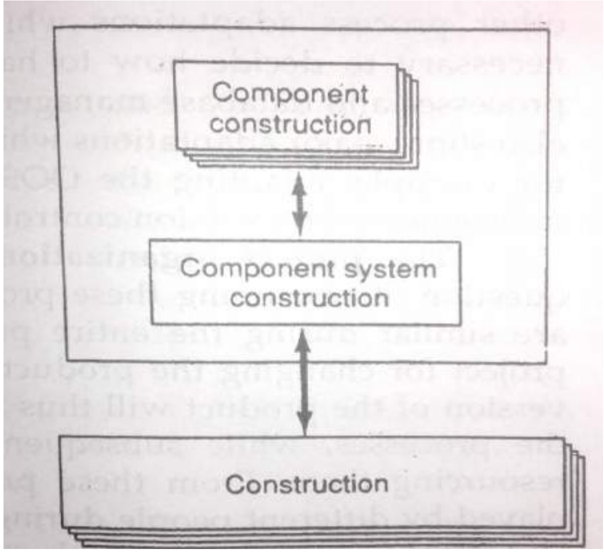


Fig: the component

process interacts with several construction process

Project organization and management

- A necessary but not sufficient, condition for successful software software development is good project management.
- A project is divided into numbers of milestones and the managerial and technical aspects must fit together to achieve this milestones.
- Milestones are concrete, objectively defined deliverable.
- Milestones are often combined with reviews with audits of the work done so far. This division aims to give better control of the projects.

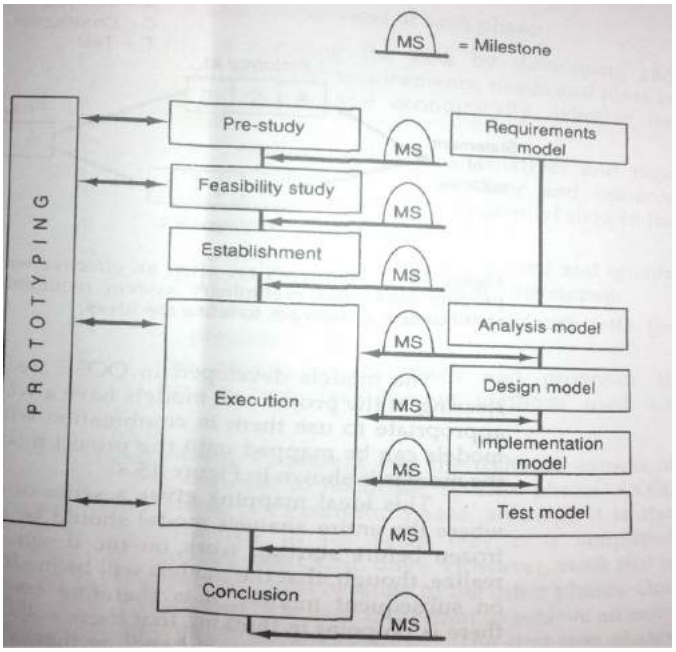


Fig: Management part of the project

- Project management phases are :

  i. **Pre-study:** It studies about if the project is practicable or not. It is done by defining and evaluating different kinds of requirements to judge projects technically and practically.

  ii. **Feasibility Study:** It studies different technical alternatives and their consequences.

  iii. **Establishment:** Detailed plans and resource plans are developed.

  iv. **Execution:** The projects is developed in accordance with the plans previously prepared.

  v. **Conclusion:** The project is completed and proposals to improve the project and development methods are summerized.

Project Staffing:

One of the difficulties in software development lies in the staffing problems. Software development is an interdependent group task. A group of people with different knowledge and skills, which we call a software project team, work together to develop software. Accordingly, the project team influences the outcome of software development. Therefore, project staffing, that is, how to form software project teams, has persistently been a key question of software organizations.

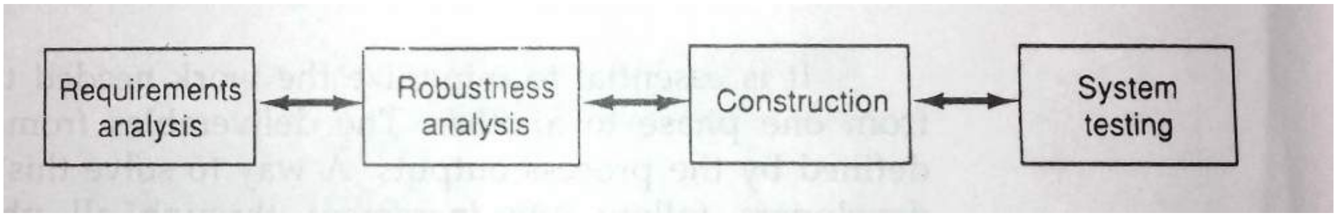


Fig: The coordination process in the OOSE

Different development group can form for a projects:

- **System architecture group:** These people are responsible for making system architecture and delivering coherent idea to the projects. They are core of development and they same for at least first three processes. i.e. requirement analysis, robustness analysis, construction. Project manager belongs to this groups.
- **Requirement analysis group:** Initial requirement analysis is done by a small group with interaction with end users.
- **Development personnel group:** More detail work should be done by development personnel who are skilled for their activity. It is good to have same person for the same group of objects in all activities. For example: a person specifying a

particular use case should also offer object, use case design and implementation block for that use case.

- During construction phase more people can add to existing group or add new group to manage more people needed in this phase
- **Testing group:** Testing is done in a separate phase often by separate group.
- Besides the actual development groups, there are other roles and groups in the projects:
    - Methodologist
    - Quality Assurance
    - Documentation, manuals and training
    - Reuse coordinator
    - Staff
    - Help system coordinator

Software Quality Assurance

i. Software quality assurance(SQA) aims that the final product will have an acceptable quality.

ii. Cost and time are often tracked in the early stages, rather than quality, but quality problems appears later in development.

iii. Quality assurance focuses on both the product and process.

iv. Documenting everything important helps to carry out quality assurance in correct way.

v. The main tools for quality assurance are development process, reviews and audits, testing and metrics.

vi. To achieve good quality disciplines, and high quality awareness, an independent quality group responsible for quality assurance in the development department may be needed.

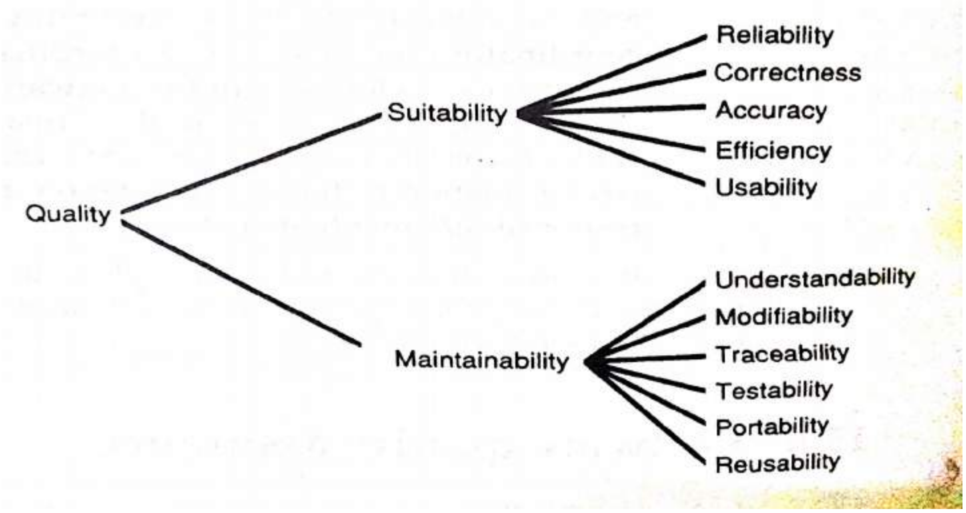


Fig: Some characteristic of software product quality

Software metrics

- Software metrics is a necessary method for controlling a development, which can measure either the process of development or various aspects of the products.
- Process-related metrics measures things like total development time, schedule time and no .of faults during testing.
- Example of process related metrics are:

- Total development time,

- development time in each process and subprocess

- cost for quality assurance.

- Process related metrics may form a basis for future planning.
- Product related metrics has not been demonstrated to be a usefull quality predictor.
- The actual code metrics that are more appropriate for OOSE are:
  - Total no. of classes
  - Number of classes reused and the number newly developed.
  - Total number of operations.
  - Number of operation reused and the number newly developed etc.
- Some statistical metrics interesting to measure are:
  - average number of operation in a class.
  - Length of operations.
  - Stimuli sent from each operation.

- Average number of inherited operation.
- For instance, the mcCabe cyclomatic complexity measures the complexity of graph such that it draws the sequence of program as a graph. N= Connection – Nodes + 2

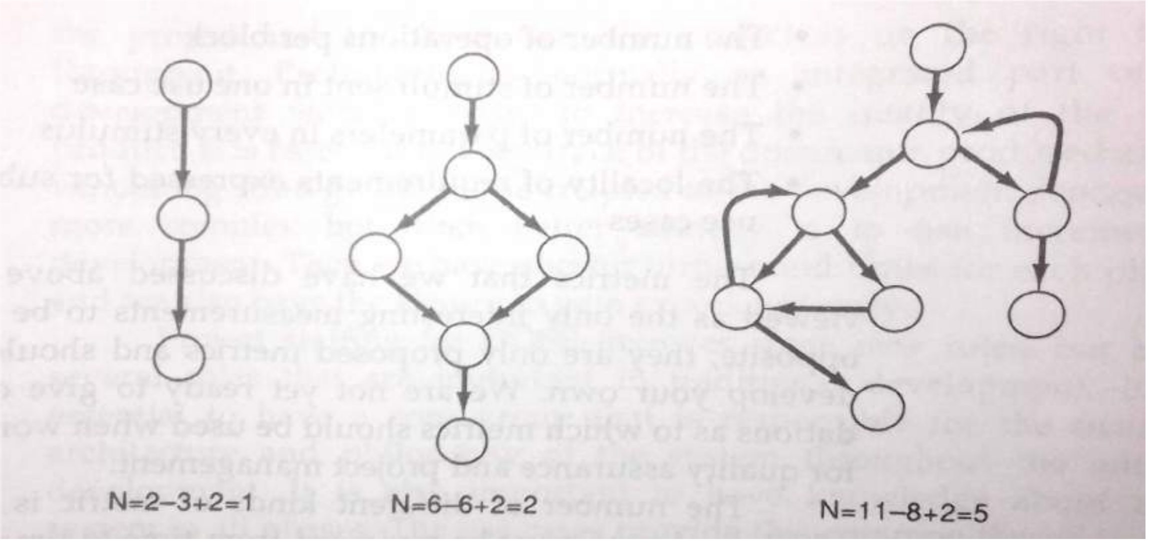


Fig: McCabe Complexity Metrics

Unit 5

Object Oriented Analysis/Coad-yourdon(OOA):

- In OOA, an analysis model is developed to describe the functionality of the system.
- The idea in Coad-yourdon design is to extend this model with respect to processes(tasks), human interfaces and DBMS.
- This methods consists of five steps:

i. **Finding class and objects:** Specifies how class and object should found.

ii. **Identifying structure:** It is done in two different way:

   - generalization-specialization(gen-spec)
   - whole part structure

iii. **Defining subjects:** It is done by partitioning the class and objects model into larger units. Subjects are group of class and objects

iv. **Defining attributes:** It is done by identifying information and the association that should be associated with each and every instance. The identified attributes are placed in correct order of inheritance hierarchy.

v. **defining services:** it means defining operations of classes.

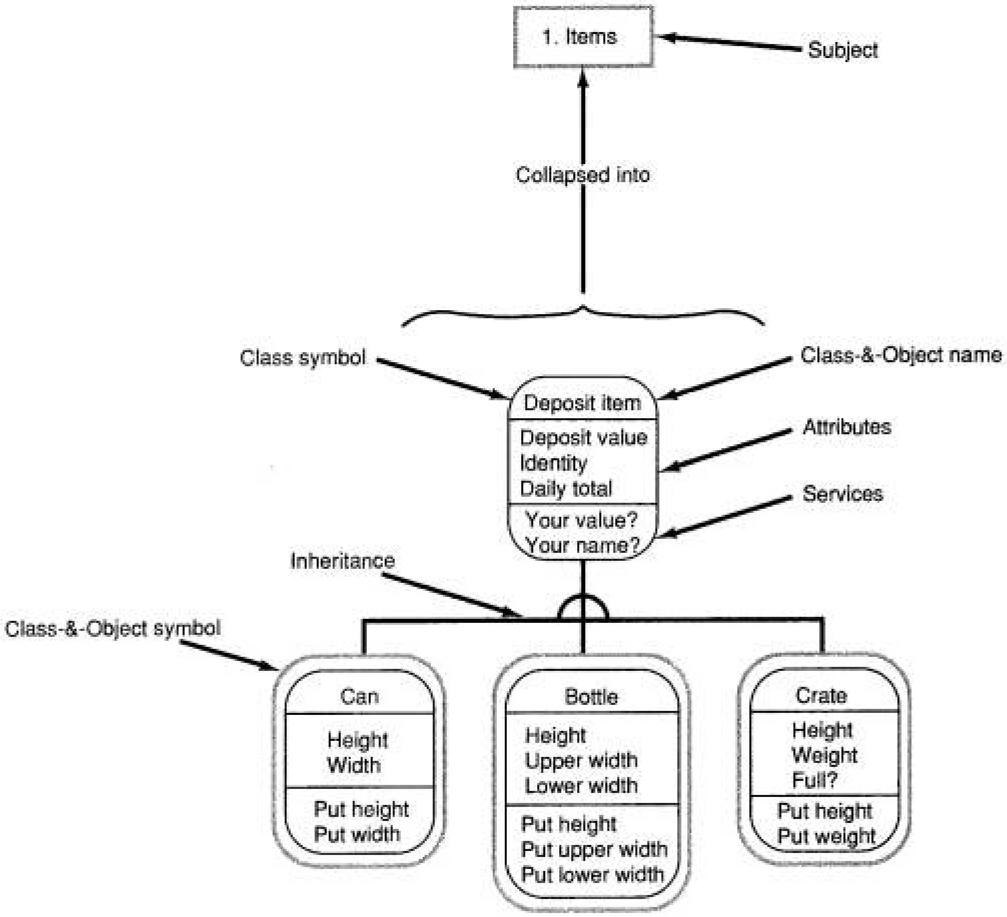
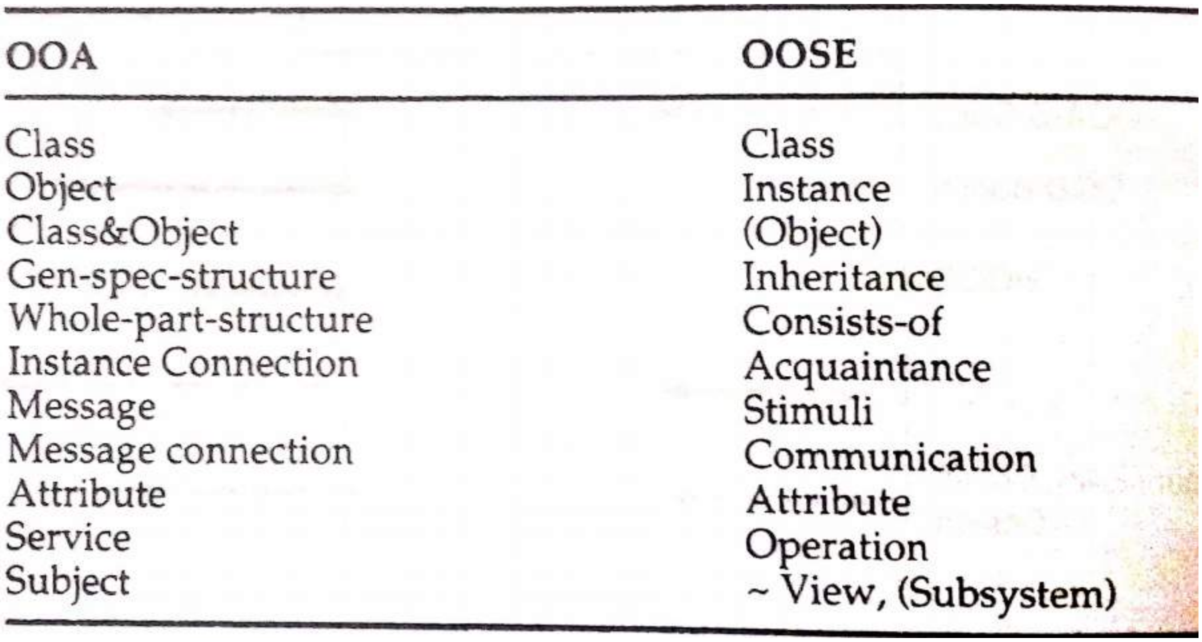


Fig: OOA applied to part of recycling system



OOA	OOSE
Class	Class
Object	Instance
Class&Object	(Object)
Gen-spec-structure	Inheritance
Whole-part-structure	Consists-of
Instance Connection	Acquaintance
Message	Stimuli
Message connection	Communication
Attribute	Attribute
Service	Operation
Subject	~ View, (Subsystem)

Object Oriented Design(OOD/BOOCH):

- It is a widely used object oriented method that helps us design our system using the object paradigm
- It covers the analysis and design phases of an object oriented system.
- The Booch method consists of the following diagrams :

  i. Class diagrams,

  ii. Object diagrams,

  iii. State Transition diagrams,

  iv. Module diagrams

  v. Process diagrams,

  vi. Interaction programs.
- Methods involves following steps:

  i. **Identify classes and objects:** Involves finding key abstraction in problem space and important mechanisms that offer the dynamic behavior over several objects.

  ii. **Identify class and object semantics:** involves establishing the relationship between classes and objects identified earlier.

  iii. **Identify class & object relationships:** involves extending the previous activities to include the relationship between classes and objects and to identify how these interact with each other.

  iv. **Implementing classes and objects:** involves delving into classes and objects and determining of how to implement them. A decision is made on how to use particular programming language to implement this classes.

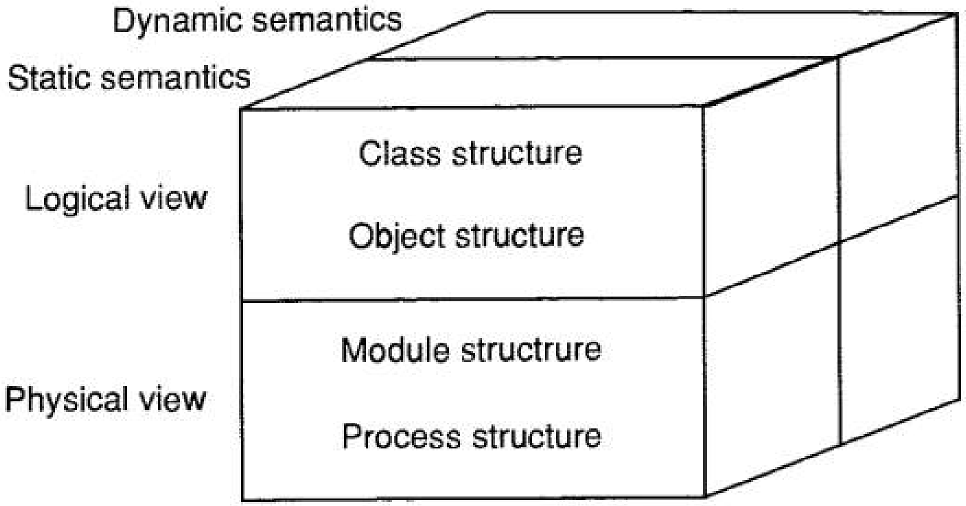
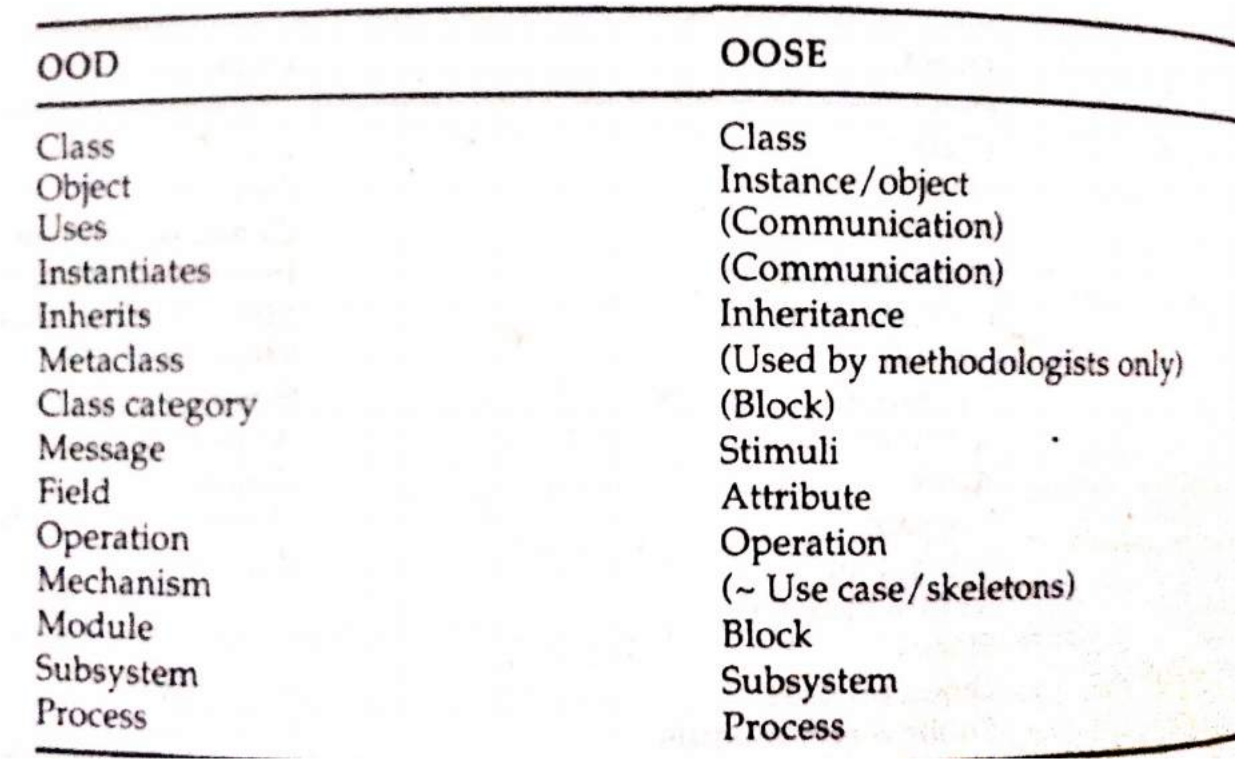


Fig: Documentation aspects in OOD



OOD	OOSE
Class	Class
Object	Instance/object
Uses	(Communication)
Instantiates	(Communication)
Inherits	Inheritance
Metaclass	(Used by methodologists only)
Class category	(Block)
Message	Stimuli
Field	Attribute
Operation	Operation
Mechanism	(~ Use case/skeletons)
Module	Block
Subsystem	Subsystem
Process	Process

Hierarchical Object Oriented Design(HOOD)

- HOOD is a method of hierarchical decomposition of the design into software units based on identification of objects, classes and operations reflecting problem domain entities.
- It is a detailed design method which Starts after analysis & Extends down to coding and testing.
- It has Object Oriented Paradigms.

  i. Unit of decomposition is no more the action, but the object

  ii. An object is an abstraction of a real world object.
- The hierarchy described in HOOD takes two forms:

  i. Uses: dependence of one object on another's services

  ii. Functional decomposition: object split into child objects, to give functionality of the parent object
- this method has four phases:

  i. Problem definition: A statement of the problem is made to provide a context for the current object level.

  ii. Development of informal solution strategy: solutions to the problem defined previously is outlined.

  iii. Formalization of the strategy: major concept of the informal solution strategy is extracted to formalize the solution.

  iv. Formalization of the solution: It is done by developing a formal model of each identified objects. This is performed in five steps:

i. **identification of objects** is done by extracting the nouns from the informal solution strategy and selecting appropriate ones.

ii. **Identification of Operations** is done by extracting the verbs from the informal solution strategy.

iii. **Grouping objects and operations** involves attaching each operation to an appropriate object.

iv. **Graphical description** is done by using HOOD graphical formalism.

v. Justification of design decision are performed by designer, who explains the reason for his decision.

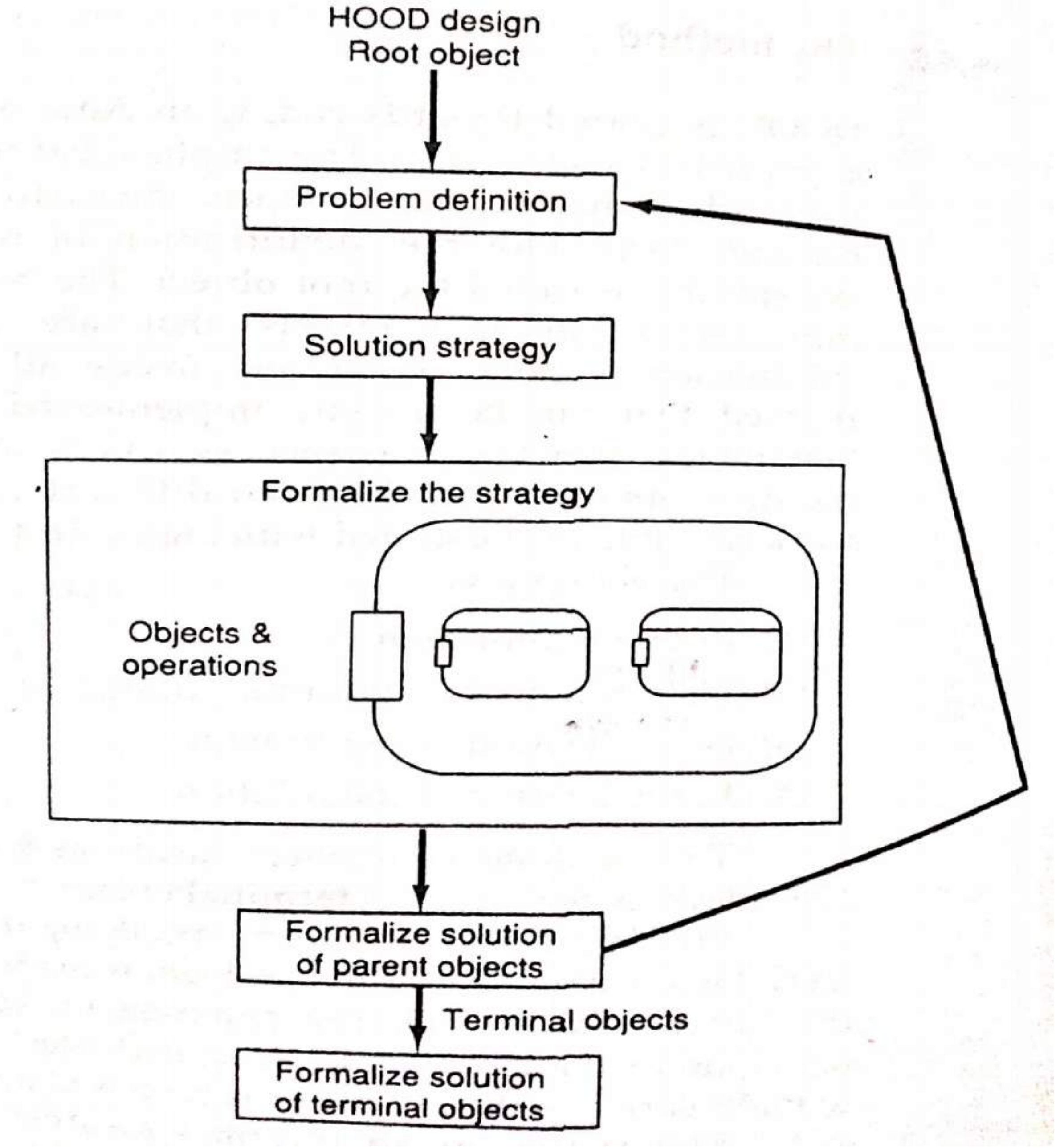


Fig: basic design steps in HOOD

Object Modeling Technique(OMT):

- The Object modeling technique covers analysis, design and implementation of a system using an object-oriented technique.

- OMT is a fast, intuitive approach for identifying and modeling all the objects making up a system.
- Details such as class, attributes, method, inheritance and association also can be expressed easily.
- OMT consists of four phases, which can be performed iteratively.
  1. **Analysis:** The results are objects and dynamic and functional models
  2. **System Design:** The results are a structure of basic architecture of system along with high-level strategy decisions.
  3. **Object Design:** This phase produces a design document, consisting of detailed objects static, dynamic and functional models.
  4. **Implementation:** This activity produces reusable, extensible and robust code.

OMT separates modeling into three different parts.

i. **An object model:** describes the static structure of the system with class and relationship.

ii. **A dynamic model:** Captures temporal aspect of the object model with events and state of the objects. It is presented by the state diagrams and even flow diagrams.

iii. **A functional model:** describes the computation in terms of how output values are derived from input values. It is presented by data flow and constraints.

- **Deliverable:** It consists of three model

i. Object Model

ii. Dynamic Model

iii. functional Model

Responsibility Driven Design(RDD):

- RDD is a way to design that emphasizes behavioral modeling using objects, responsibilities and collaborations.

- In a responsibility-based model, objects play specific roles and occupy well-known positions in the application architecture.

- Each object is accountable for a specific portion of the work. They collaborate in clearly defined ways, contracting with each other to fulfill the larger goals of the application.

- By creating a "community of objects", assigning specific responsibilities to each, you build a collaborative model of our application.

- This methods comprises a number of phases where each phase is described as number of activities.

- The **exploratory phase** consists of

i. **Classes:** They are found by reading the specification and extracting the essential nouns,

ii. **responsibilities of each class:** by looking verbs in the requirements specification we can find the action of objects in the system.

iii. **Collaboration between objects:** the actual collaboration between the object is found by asking question like "with what does this class need to collaborate to fulfill its responsibilities ?"

- The **refining phase** consists of:

  i. **Hierarchical between classes:** abstract classes can be extracted, inheritance hierarchies between classes are further refined.

  ii. **Subsystems:** groups of classes collaborate to fulfill their responsibilities.

  iii. **Protocol:** This responsibilities and contracts are further refined into protocols which shows the specific signature of each operations.

- The output of RDD is a design specification consisting of

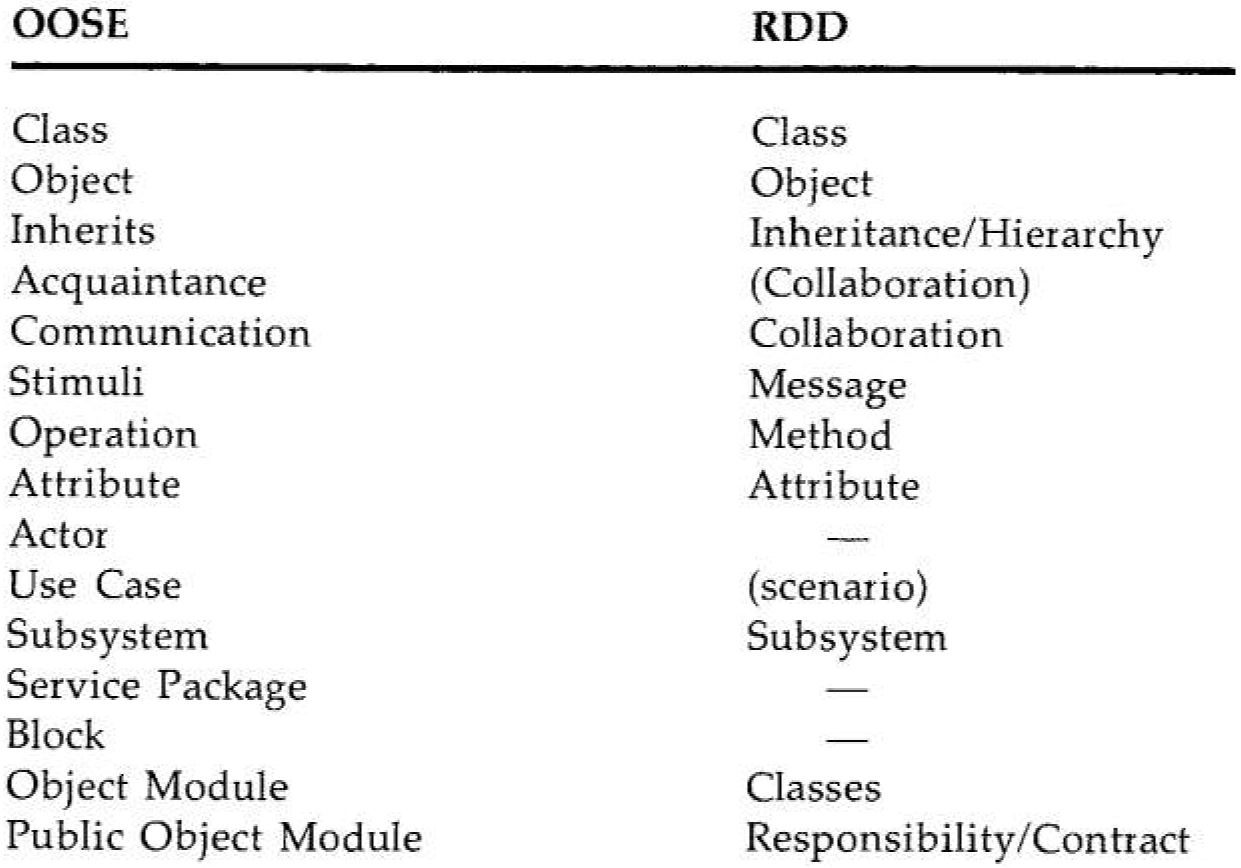
  i. A graph of each class hierarchy.

  ii. A graph of the collaboration for each subsystem.

iii. A specification of each class.

iv. A specification of each subsystem.

v. A specification of the contacts supported by each class and subsystem.



OOSE	RDD
Class	Class
Object	Object
Inherits	Inheritance/Hierarchy
Acquaintance	(Collaboration)
Communication	Collaboration
Stimuli	Message
Operation	Method
Attribute	Attribute
Actor	—
Use Case	(scenario)
Subsystem	Subsystem
Service Package	—
Block	—
Object Module	Classes
Public Object Module	Responsibility/Contract

Fig:The concept of OOSE related to RDD