OBJECT ORIENTED SOFTWARE ENGINEERING

⇒ what is object orientation?

⇒ Software Development Life Cycle

↳ It only looks at the software component's development, planning technical aspects, software quality testing and deployment of working software.

↳ Is about building software "only" in a phased approach systematically

⇒ System Development Life cycle

↳ It involves end-to-end people, process, software / Technology deployment. Includes change management, training, Organizational updates

⇒ OOSE

→ Is a technique that is used in software design in object-oriented programming

→ Specific aim to fit the development of large, real-time systems.

→ It is first ~~design~~ object-oriented design methodology the employs use cases in software design.

→ Objectory is built around several different Models:

↳ Use-case model: How user interact with use cases.

↳ Domain object Model: Objects of real world are mapped into the Domain object Model.

↳ Analysis of object Model: represents how the source code is to be implemented.

↳ Implementation Model: represents implementation of system

↳ Test Model: constitutes test plans, specifications and reports.

→ OOSE is one of the precursors of the Unified Modeling Language (UML) such as Booch and OMT.

→ It includes requirements, analysis, design, implementation and a testing model.

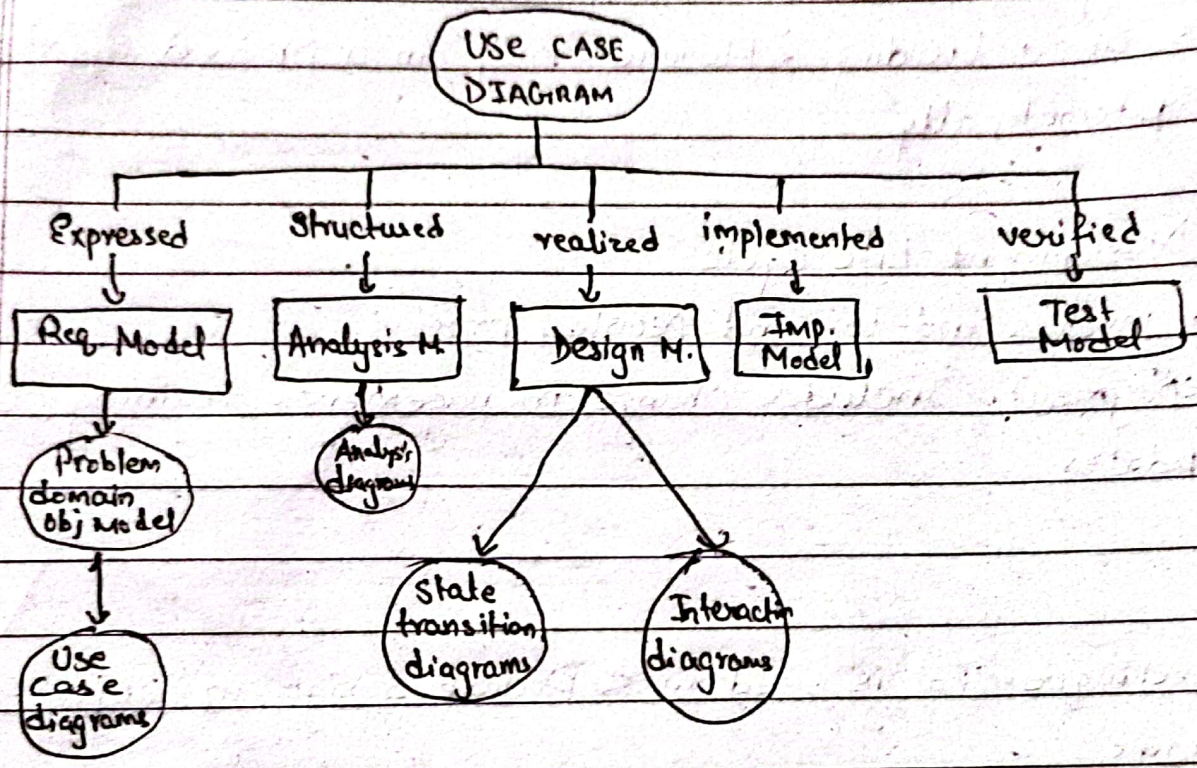


fig: Object-Orient SE

⇒ Object Oriented Analysis:

→ This phase of software development is concerned with determining the system requirement & identifying their classes and their relationship to other classes in problem domain.

→ The goal is to find domain of the problem and system's responsibilities by knowing all user's needs which is.

accomplished by models.

→ These models concenterate on describing what the system does rather than how it does it.

→ Object oriented analysis has following steps:

↳ Identifying the actors

↳ Develop a simple business process using UML activity diagram.

↳ Develop the Use case.

↳ Develop interaction diagrams.

↳ Identify classes

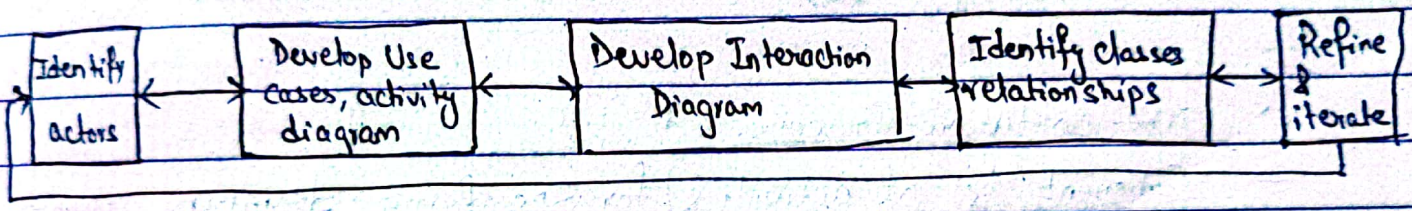


Fig: OO analysis

⇒ OO Design:

→ Aim is to design the classes identified during the analysis phase.

→ Here, we identify and define additional objects, classes that support implementation of requirements.

→ OOD is highly incremental,

→ Here, we first build the object Model based on objects & their relationships, then iterate and refine the model such as:

↳ Design and refine classes

↳ " " " attributes

↳ " " " Methods

↳ " " " Structures

↳ " " associations

→ Guidelines in OOD

↳ Reuse, rather than rebuild, a new class.

↳ Design a large no. simple classes, rather than a small no. of complex classes.

↳ Critique what you have proposed

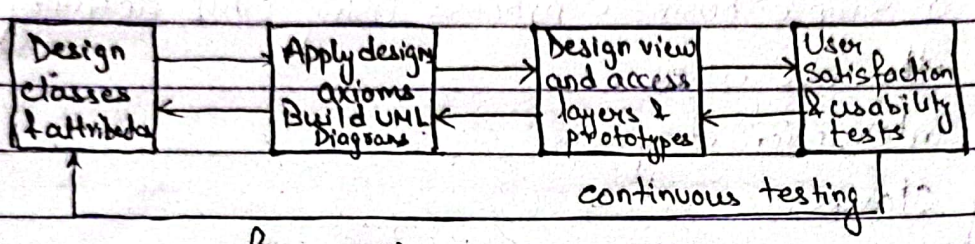


fig: OOD

⇒ The Software Process

→ Coherent set of activities for specifying, designing, implementing and testing software systems.

→ A structure set of activities required to develop a software system.

↳ Specification

↳ Design

↳ Validation

↳ Evolution

→ Software process model is an abstract representation of a software process.

→ It presents a description of a process from some particular perspective.

=> SDLC Models / Software process Models.

i) Waterfall Model

-> Requirement are well documented or known.

-> Technology is understood and not dynamic.

-> Project is short.

=> Object Oriented System Development

-> System development refers to all activities that go into producing an information systems solution. These activities consists of

↳ Requirement analysis

↳ Modeling, Design, implementation, testing and maintenance.

-> According to Niklaus Wirth – A software system is a set of mechanism for performing certain actions 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

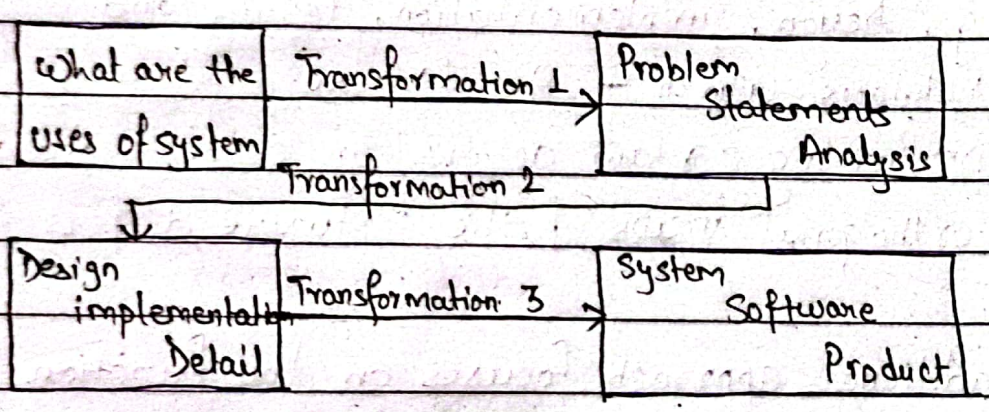
↳ Object Oriented system development centers on the object, which combines data and functionality.

-> Object Oriented System Development is a way to develop software by building self-contained modules or objects that can be easily replaced, modified and reused.

⇒ Object Oriented System development can be divided into five phases.

- Analysis
- design
- Implementation
- Testing
- Refinement

→ Can be viewed as a series of transformations, where the output of one transformation becomes input of the subsequent transformation.



⇒ Function / Data Method

→ In this method, function is active and have behavior and data is passive information holder which is affected by function.

→ The system is broken down into functions whereas data is sent between those functions.

→ System developed using function / data often becomes difficult to maintain
→ System generated using this method is often unstable, a slight modification will generate major consequences.

⇒ Model Architecture

→ System Development is basically concerned with developing models of the system i.e. identifying 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 fit perfectly for one model but totally wrong in another model
→ Important criteria is that it must be robust against modification and should help to understand system. 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:

→ Defines the system boundaries and functionality that should offer.
→ functions as contact between developer and the orderer of the system.
→ Should also be readable for none OOSE practitioners.
→ This model govern the development of all other model so it 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
- ↳ Tested in testing Model.

→ Requirement Model consist of

↳ Use Case Model
- → Is a specific way of using the system by performing some part of functionalities.
- → Specifies the interaction between the actor and the system.

↳ Interface Description
- → Model must be presented in a way which is understood by clients.

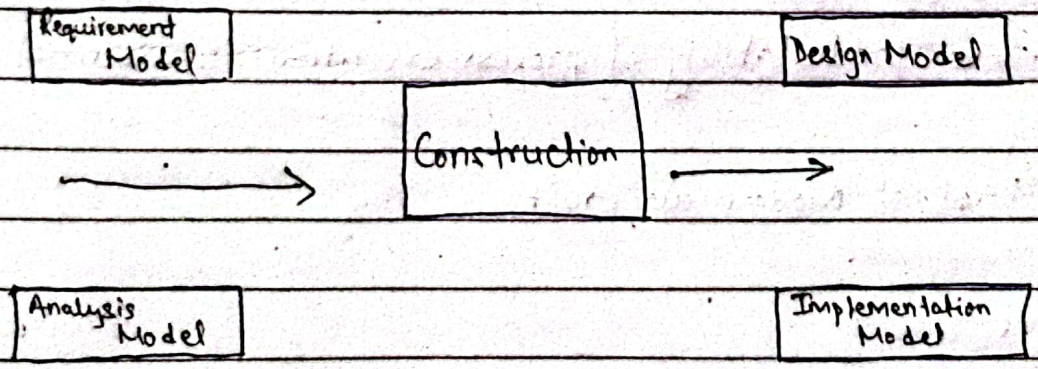
↳ Problem Domain object.
- → when requirements specification exists in very vague form, it is difficult to define task and boundary of a system.
- → It is good to develop a logical view of the system using problem domain object.

=> Three main reasons for having construction phase.

i) Analysis Model is not Sufficiently formal.
   ↳ To change source code we must ~~define~~ refine the object.

ii) The actual system must be adopted to implementation variable
   ↳ In analysis we assumed ideal world for our system. Actually there is no ideal world and we must adopt to the environment in which the system is to be implemented.

iii) To validate the analysis results:
   ↳ In construction we see the results of analysis.
   ↳ If we discover unclear implementation model, the construction model is further divided into two phases, Design and implementation
   ↳ The design model is a further refinement & formalization of analysis model where consequences of implementation environment have been taken into account



⇒ 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.

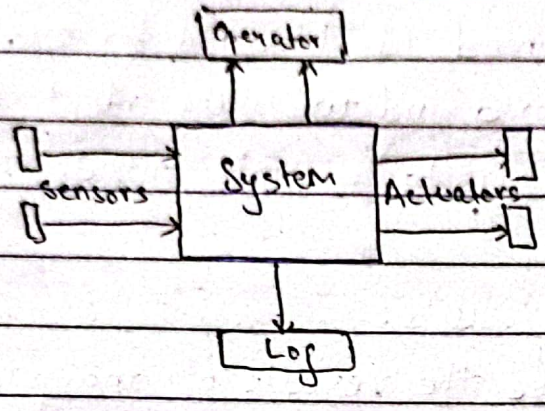


Fig: Abstract Model of RTS

→ RTS has means of observing what's going on, Controlling processing from operator & keeping history via logging

⇒ Hard RTS

→ System with hard deadline that must meet otherwise a catastrophe can occur

→ Deterministic predictability of process execution is essential property

→ Eg: Control of modern air craft

⇒ Soft RTS

→ System where services are provided in real-time but catastro-phe will not occur if immediate service is not provided.

→ Execution of resources are stochastically distributed based upon the quantities of resources available & loading of system.

→ Eg: Digital telephone exchange.

⇒ Component:

→ Is a standard building unit in an organization that is used to develop application.

→ Make it reusable

→ Must include all reasonable operations that make it meaningful to use for any other developers

→ Should provide general abstraction so that it can be widely used

→ Components that one can modify according to one's need or component that needs to be modified before reusing is called white box component.

→ Components that doesn't need to modify for use or components which are designed to solve specific domain problem are Black box components.

⇒ Use of Components:

→ General Entity Object: an entity object that is used to develop other entity object & whose information should be stored in db.

→ Interface object: Eg: windows system, windows button, text field etc.

→ Control objects: General function such as logging activities, data Collecti

There are two types of activities for component management.

i) One for the design of complete component system

ii) One for design of individual components.

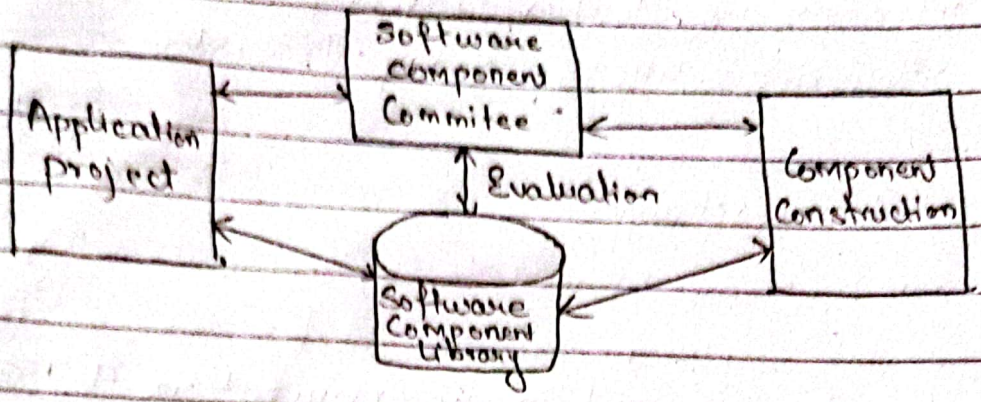


Fig: An organization of component management.

⇒ Managing Object-oriented Software Engineering

→ key factors:

↳ Project Selection and preparation
↳ Product development Organization
↳ Project organization and management
↳ Project Staffing
↳ Software quality assurance
↳ Software metrics.

⇒ Project Management Phases:

↳ Pre-Study
↳ Feasibility Study
↳ Establishment
↳ Execution
↳ Conclusion

⇒ Component Based Software Engineering:

↳ Is a process that focuses on the design and development of computer-based systems with the use of reusable software components.

↳ This practice aims to bring about an equally wide-range of benefits in both short-term and long-term for the software and for organizations that develop such software.

↳ It is a set of pre-built standardized software components that is made available to fit in a specific architectural styles

↳ Its approach is based on the idea to develop software system by selecting appropriate components and then to assemble them with a well built architecture.

↳ Objectives:

i) Reduction of cost and time for building large and complicated system.

ii) Improving the quality of software

iii) Detection of defect within the system.

Eg:

→ If someone want to build a stereo system by assembling the compo-nents like sound box etc. then he might be experiencing some advantage than those who build the system from the basic. The main concept is that, those components need not be changed for their respective purposes.

=> function/data Oriented SE. (traditional)

↳ Tradional SE was used to develop softwares in a procedural manner.

↳ The traditional approach includes 8 main phases

i) Requirements
ii) Analysis
iii) Design
iv) Specification
v) implementation
vi) Testing
vii) Deployment
viii) Maintainance

↳ This approach uses traditional projects and leads software developers to focus on decomposition of larger algorithms to smaller ones.

↳ Depends on the size of project and type of projects. It might take large duration for large projects

↳ Complex

↳ followed a linear sequential model

⇒ OOSE

↳ System is viewed as set of objects

↳ Each ~~ma~~ Different OO Methodologies include:

↳ philosophy behind each phases.

↳ Workflows and individual activities within each phase.

↳ UML diagrams, texual description and code (artifacts)

↳ Dependencies bet^n the artifacts.

↳ Notations for Different kinds of artifact.

↳ The OOSE Method includes 5 phases and the system is viewed as a set of objects.

i) Analysis

ii) Construction — Includes 3 traditional ~~set~~ approaches.

iii) Testing

iv) UML

v) Maintenance

↳ Requires More time than traditional approach and that leads to more cost.

⇒ COAD - Yourdon's Method of Object Oriented Analysis.

↳ Has its ~~at~~ primary Strength in System analysis

↳ This Method is based on "SOSAS" which Stands for five Steps that help make up the analysis part of their methodology

Step 1 : Subjects:

↳ which are basically data flow diagrams for objects

Step 2: Objects:
↳ where they identify object classes and the class hierarchies.

Step 3: Structures:
↳ where they decompose structures into two types
i) classification Structure:
↳ Handles the inheritance connection between related classes
ii) Composition Structure:
↳ Handles all of the other connection between related classes.

Step 4: Attributes
↳ Is some data or state information for which each attribute in a class has its own value.

Step 5: Services
↳ where all of the behaviors or methods for each class are identified

⇒ Following analysis, Coad and yourdon define four parts that make up the design part of their methodology.

Step 1: The problem domain component:
↳ This will define the classes that should be in the problem domain

Step 2: The human interaction component:
↳ defines the interface classes between objects.

Step 3: The task management component:
↳ This is where system-wide management classes are identified.

Step 4: The data management component:
↳ Identifies the classes needed for database access methods

Eg:

⇒ Object Modeling Technique:

↳ Is a real world based modeling approach for software modeling and designing.

↳ It was developed basically as a method to develop object-oriented systems and to support object-programming

↳ one of the most popular object oriented development techniques. Was developed by James Rambaugh.

Purpose:

→ Test physical entity before ~~testing system~~ constructing them.
→ Make communication easier with customers.
→ present information in an alternate way i.e. visualization
→ Reduce complexity of system
→ Solve real world technique

=> OMT Models

i) Object Model:

↳ Encompasses the principle of abstraction, encapsulation, modularity, hierarchy and persistence.

↳ It emphasizes on object and classes

↳ Main concept related with object Model are classes and their association with attributes.

ii) Dynamic Model:

↳ Involves states, events and state diagram on the Model.

↳ Main concept related with Dynamic Model are states, transition between states and even that triggers transition.

iii) Functional Model:

↳ focuses on how data is flowing, where data is stored and different processes.

↳ Main concept involved in functional model are data, data flow, data store, process and actors.

Phases:

→ Analysis:

↳ involves preparation of precise and correct modelling of the real world problems

↳ Starts with setting a goal i.e. finding problem statement

↳ problem Statement is further divided into OMT Models.

→ System design:

↳ Determines all system architecture, concurrent task and data Storage.

↳ High level architecture of the system is designed during this phase

→ Object Design:

↳ Concerned with classification of object into different classes and about attributes and necessary operations needed

→ Implementation:

↳ Converting prepared design to software.

↳ Design phase is converted into implementation phase.

⇒ Responsibility Driven design (RDD):

↳ Is a design ~~tech~~ technique in OOP, which improves encapsulation by using the client-server model.

↳ Is in direct contrast with data-driven design, which promotes defining the behavior of a class along with the data that it holds

↳ According to Rebecca ~~Wir~~ Wirfs-Brock, RDD have 3 main principles.

i) Maximize Abstraction

ii) Distribute behavior

iii) Preserve flexibility design

Steps in RDD

i) Find the classes in our system

ii) Determine the responsibilities of each class

iii) Determine how objects collaborate with each other to fulfill their responsibilities.

iv) Factor common responsibilities to build class hierarchies

i) Defining ~~problems~~ Responsibilities:

→ is crucial to have a good software design.

→ Use-cases are a good starting point.

→ Role is some collection of related tasks that can be bundled to a single responsibility.

⇒ Responsibilities are of two types:

↳ Doing:

→ Doing something itself, creating an object or doing a calculation.

→ initiating action in other objects

→ controlling and coordinating activities in other objects.

↳ Knowing:

→ knowing about private encapsulated data.

→ knowing about related objects.

→ knowing about things it can derive & calculate.

=> Hierarchial Object Oriented Design (HOOD).

↳ Is a detailed software design method. It is based on hierarchial decomposition of a software problem. It comprises textual and graphical representation of the design.

↳ It is intented for the Architectural design, Detailed design and coding for software to be developed in programming language such as ADA, C or FORTRAN as well as in OOL's such as C++, Ada95 and or Eiffel.

↳ A basic design step process is further split into four phases., thus defining a micro life-cycle for design step:

i) Problem definition (Exp)

↳ Problem statement

↳ Analysis and structuring of requirement data.

(Exp) ii) Development of Solution Strategy (Outline solution of the above described problem in terms of object)

(Exp) iii) Formalization of the Strategy

↳ Object identification

↳ Operation identification

↳ Grouping object and operation

↳ Graphical description

↳ justification of design decisions

(Exp) iv) formalization of the solution

⇒ Object Oriented Design Booch Method. Eg. with Eg.

↳ Is a widely used object-oriented method. It helps to design our system using object paradigm.

↳ Booch uses large set of symbols.

↳ 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 Diagrams

↳ Booch Methodology prescribes a Macro development and Micro development process

i) Macro Development Process

↳ Serves as a controlling framework for the micro process and can take weeks or even ~~Mott~~ months.

↳ It is interested less in the actual object-oriented design than in how well the project corresponds to the requirements set for it and whether it is produced on time.

ii) Micro development process.

↳ Each macro development process has its own Micro development process.

↳ Micro process is a description of the day to day

activities by a single (or) small group of developers.

↳ Consists of following steps

i) Identify class and objects

ii) Identify class and object semantics

iii) Identify class and object relationships.

iv) " " " interface & implementation.

Macro development Process

Conceptualization → Analysis & Development of the Model → Design or Create System Architecture

Established using context diagrams & prototypes

Describes roles and responsibilities of object using class diagram

Classes & relation about them using cla diagram

Maintenance ← Evolution/Implementation

Produces a stream of software executable releases

⇒ Steps to develop requirement model from user requirements

↳ Requirement Modelling Strategies

i) Flow Oriented Modeling

ii) Class-based Modeling

1) Flow Oriented Modeling

↳ It shows how data objects are transformed by process -ing the function.

Flow oriented elements are:

↳ i) Data Flow Model
- → graphical technique to represent information flow
- → Data Flow Diagram is a example of Data flow Model.

ii) Control flow Model
- → Large class applications require a control flow modeling.
- → The application creates control information instead of reports or displays
- → Process the information in specified time.

iii) Control Specification:
- → Represents the behavior of the system.
- → The state diagram includes States, transitions, events and activities.

iv) Process Specification:
- → Is used to describle all flow model processes.
- → Content of process specification consists of tables, UML diagrams, narrative texts. Etc.

2) Class-based Modeling:

↳ Represents the object. The system manipulates the operations.

↳ Consists of classes and objects, attributes

operations, class-responsibility - collaborator (CRS)

→ Classes:
↳ are determined using underlining each noun or noun clause and enter it into a simple table.

→ Attributes:
↳ Set of data objects that are defining a complete class within the context of problem.
↳ Eg: 'Employee' is a class and it consists of name, id, departme are the attributes

→ Operations:
↳ Define the behavior of objects.
→ ~~Atom~~ ~~Object~~ → Manipulates data like, adding, modifying deleting.
→ perform a computation
→ Monitors the objects for the occurence of controlling an event

→ CRS
↳ provides a simple method for identifying and organizing the classes that are applicable to system or product requirement.

⇒ Managing Object Oriented Software Engineering.

↳ Managing OOSE is an art and discipline of planning and supervising software projects.

↳ It is a process of managing, allocating and timing resources to develop computer software that fulfills requirements.

key factors involved in managing OOSE are:

i) Project Selection and preparation

↳ One of the biggest decisions that any organization would have to make is related to the projects they could undertake.

↳ The most viable options need to be chosen;

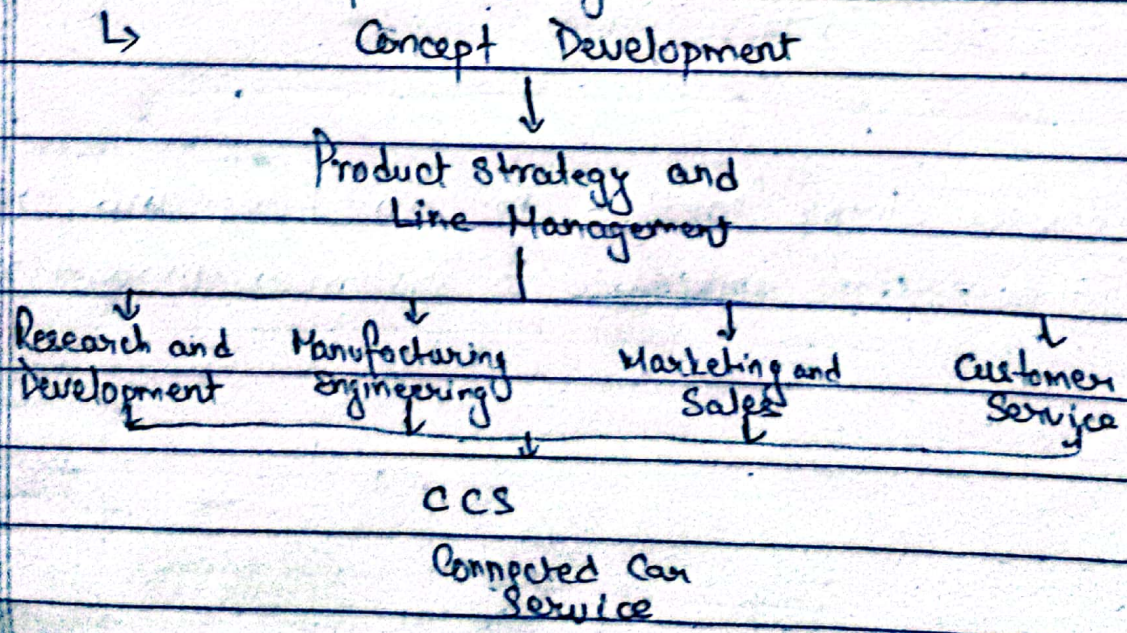
i) keeping in mind the goals and requirements of organization.

ii) Decide if a project is viable.

iii) Use appropriate project selection Methods.

ii) Product development Organization:

↳



iii) Project Organization and Management:

↳ project organization defines the relationships among resources, in particular the participants, in a project.

↳ A project organization should define decision structure, reporting structure and communication structure

↳ Phases: Pre-Study, feasibility study, Establishment, Execution, Conclusion

iv) Project Staffing:

↳ Regardless of what you do in an organization, a staff is required in order to execute work, tasks and activities.

↳ just having the required number of staff members for the project will not help you to successfully execute the project activities. (Must have necessary skills, Motivation & availability)

v) Software Quality Assurance:

↳ Is a means and practice of monitoring the Software Engineering process and methods used in a project to ensure proper quality of the Software.

↳ Is a process which works parallel to development of Software.

↳ It focuses on improving the process of development of Software so that problems can be prevented before they become a major issue.

vi) Software Metrics:

↳ Is a measure of Software characteristics which are measurable or countable

↳ These are valuable for many reasons, including measuring software performance, planning work items, measuring productivity and many other uses.

↳ Can be classified into two types:

i) Product Metrics

ii) Process Metrics

Product Metrics:

↳ These are measures of various characteristics of the software products.

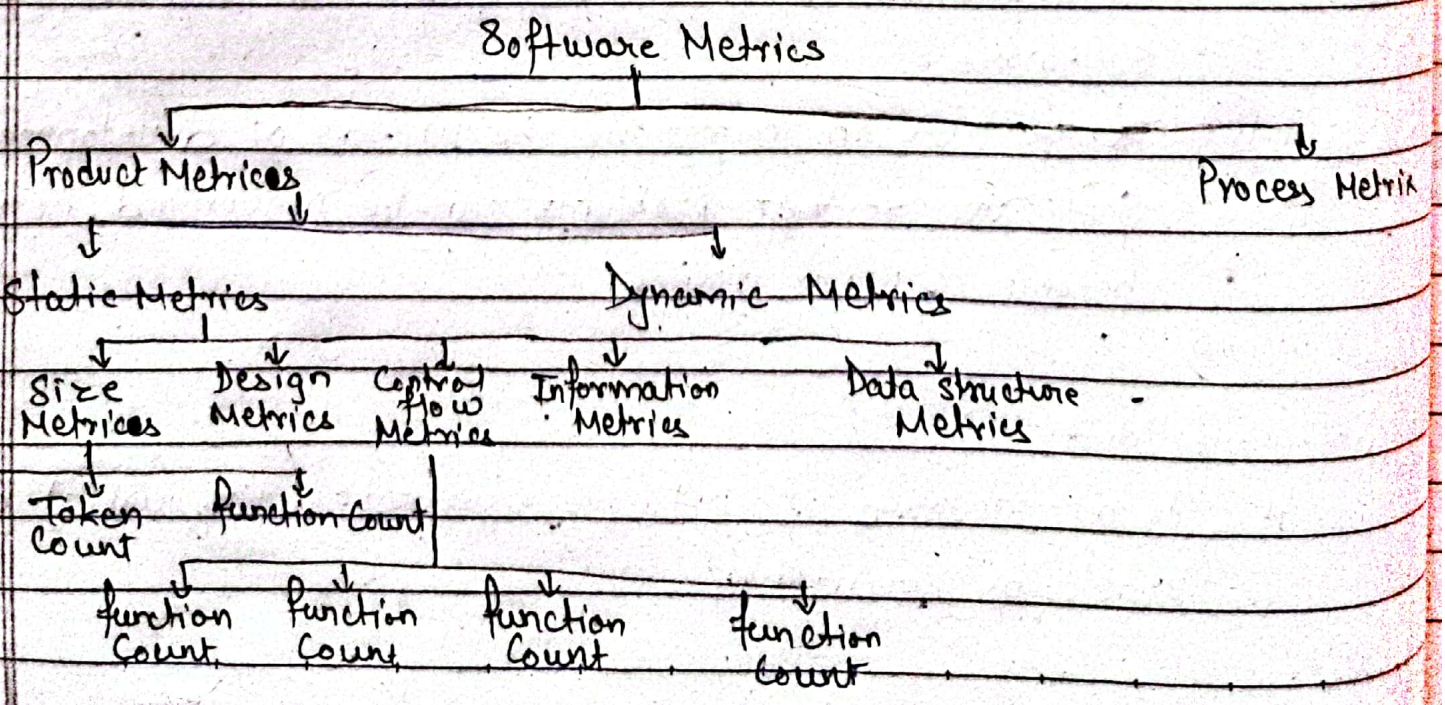
↳ Two important software characteries are:

i) Size and complexity of software.

ii) Quality and reliability of software.

Process Metrices:

↳ They are used to measure the charateristics of methods, techniques and tools that are used for developing softwares.



=> Process of 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 defined.

↳ Select people with good experience in system development who has positive view of change.

↳ Select project manager with high degree of interest in task.

↳ The staff should work full time in the projects and not be distracted by other projects.

↳ Have a detailed plan developed in advanced to create a base of work. Perform evaluation at all stages.

Preparation :

↳ All individual involved in the new order of work need education and training.

↳ A new development process involves a lots of changes which brings potential risks.

↳ These risks can be minimized by :

i) Risk identification

ii) Risk valuation

iii) Managing the risks