O.O.S.E.

Date
Page

Unit-1:

* **Software life cycle models:** (also called Process Models.)
→ structured approach to software development.
→ Provides framework for planning, Managing and controlling the process of developing a complete information system.

~~Most common Models are~~:

*

→ **They are Important because** ~~of~~
→ acts as guide to project & Meet client req.
→ helps in evaluating, scheduling and estimating deliverables.
→ provides framework for standard set of activities
→ ensures correct and timely delivery to customers.

Phases of SDLC:

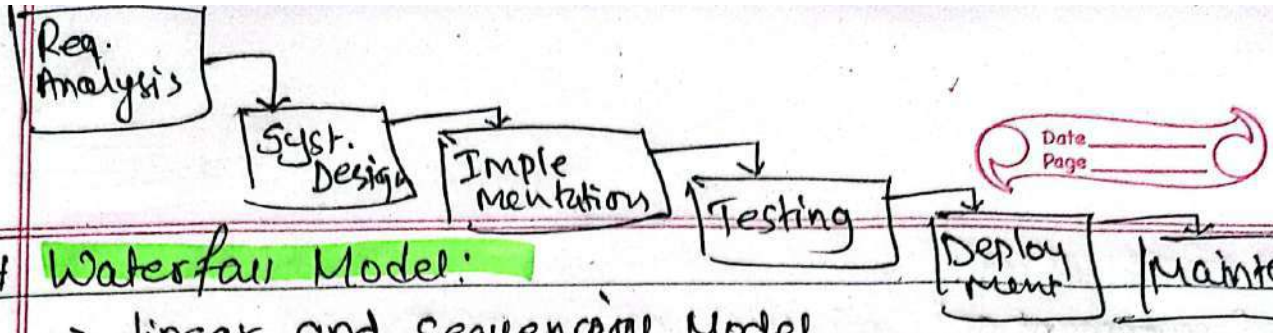
Requirement Analysis.
Defining. → SRS document created.
Designing. → High level Design, low L.D.
Coding
Testing
Deployment
Maintenance

Quality of Good SRS- document:

→ correctness.
→ unambiguousness.
→ Completeness.
→ Consistency.
→ Ranking for importance } and/or.
→ Stability rating. }
→ verifiability.
→ Modifiability.
→ Traceability.
→ understandable by consumer.

Some Common SDLC models are:

Waterfall Model
Spiral Model.
Agile Model
V-Shaped Model.
BigBang Model.
DevOps Model.
Rapid Application Development (RAD) model.
Incremental Model.
Prototyping Model.
Iterative Model.



Waterfall Model:

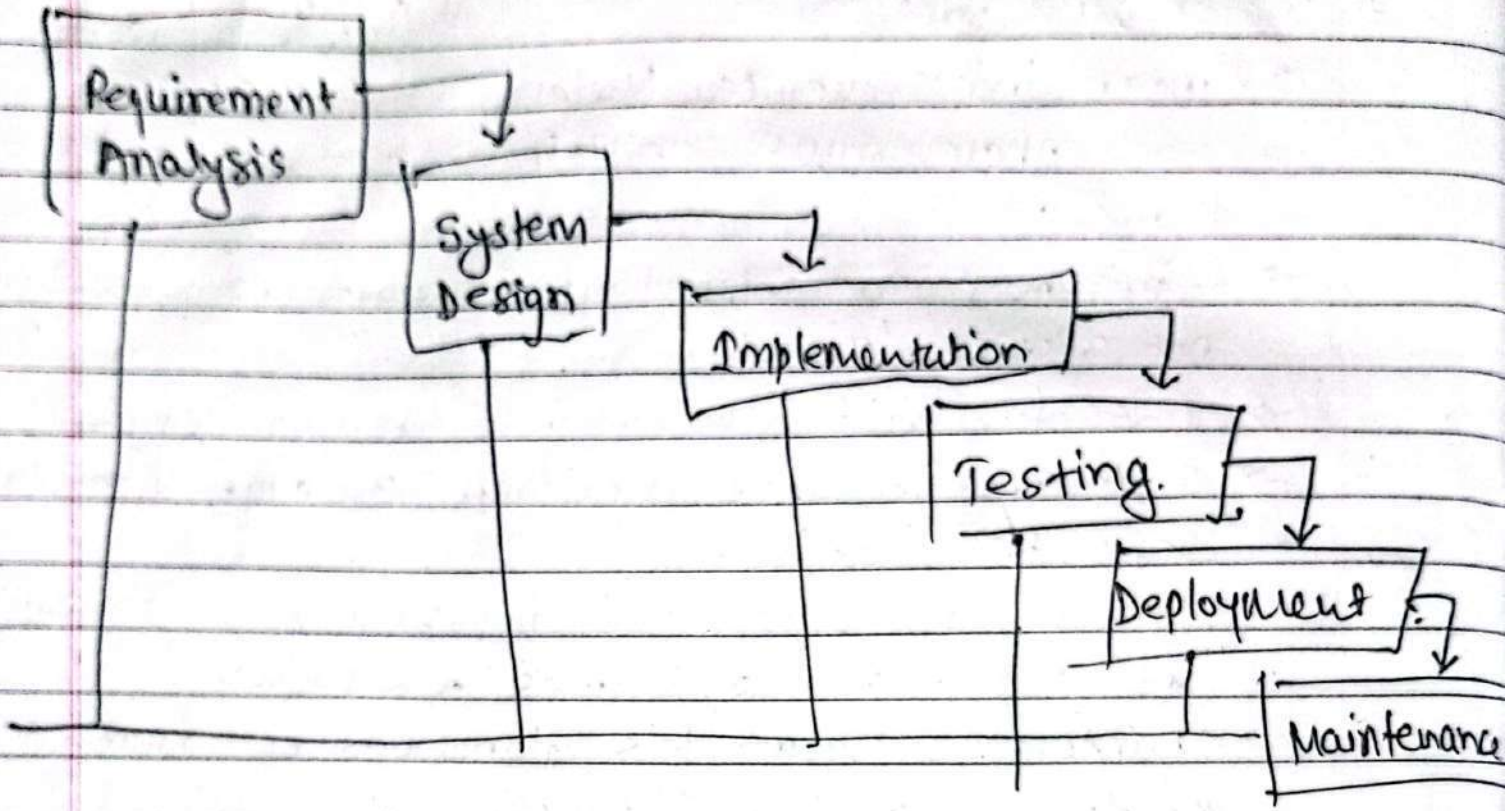
→ linear and sequencial Model
→ each phase must complete Before the next one begins.
→ Requirement are defined at the start and changes are not allowed later.
→ It is low-cost Model as there are no iterations.
→ It is easy to understand and Manage due to sequential nature.
→ It need high expertise during requirement specific as there are no changes occur later.
→ Maintenance seems less glamorous as new cycle needs to start for changes.
→ Documentation is vital as it serves as guidelines.
→ Guarantee of success is less due to inflexible to change.

Advantages:

→ simple and easy to understand.
→ well suited for small Project and clearly defined requirements.

Disad.

→ Difficult to go back to previous phases.
→ Inflexible to Changes.



Iterative Model: // gncremental Model.

→ Requirements are defined at beginning and can be changed in subsequent gteration.
→ The cost is low initially but may increases by each iteration
→ High expertise required to manage iteration.
→ Risk are gdentified and Manage in each iteration.
→ gterative cycle encourage the re-use of components.
→ less flexible compared some other Models as it is still sequential in pattern.
→ Documents updated in each iteration.
→ User may provide feedback between each iteration
→ frequent gteration incresses the chances of succes.

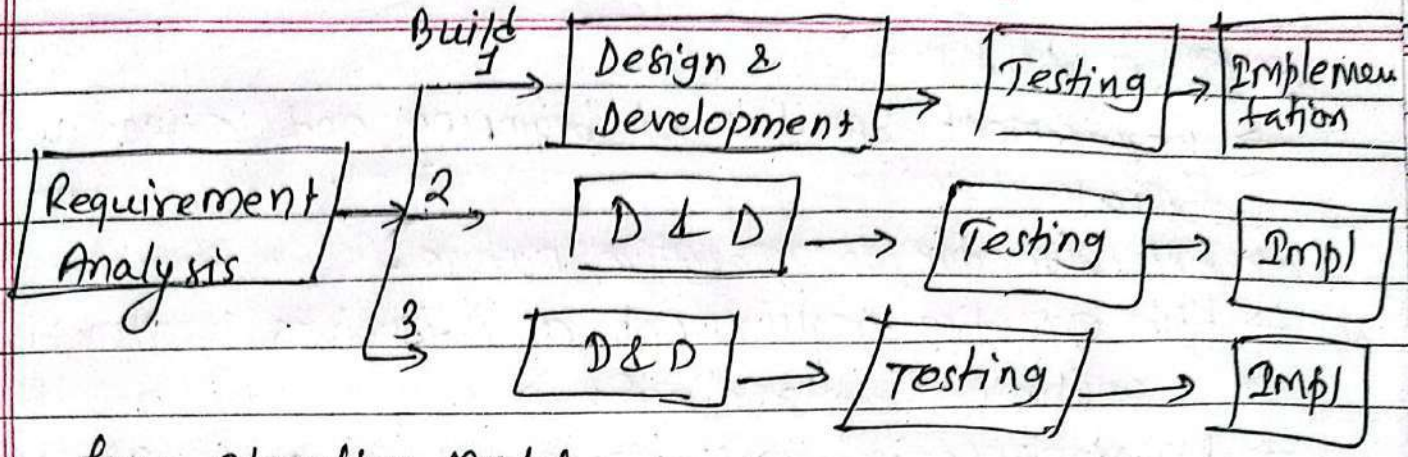
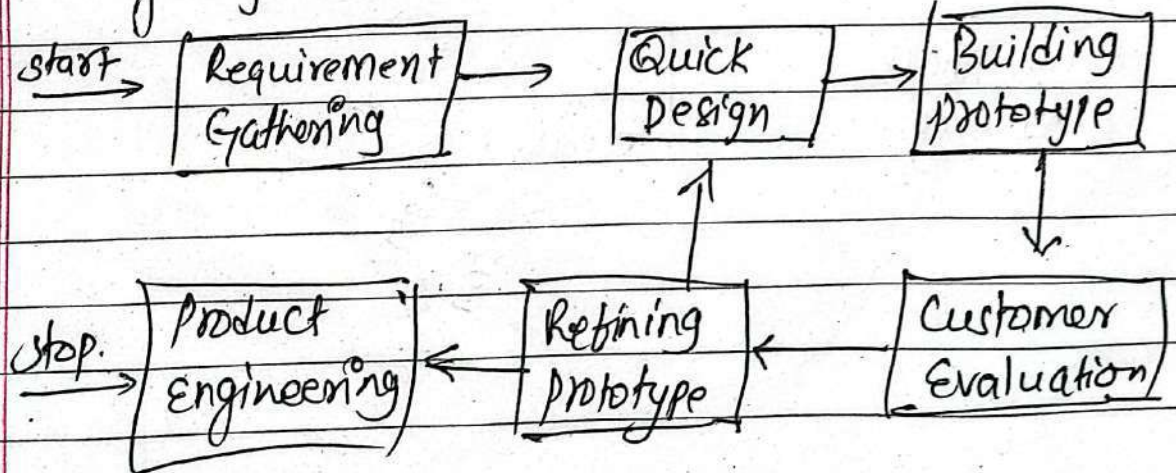


fig: Iterative Model.

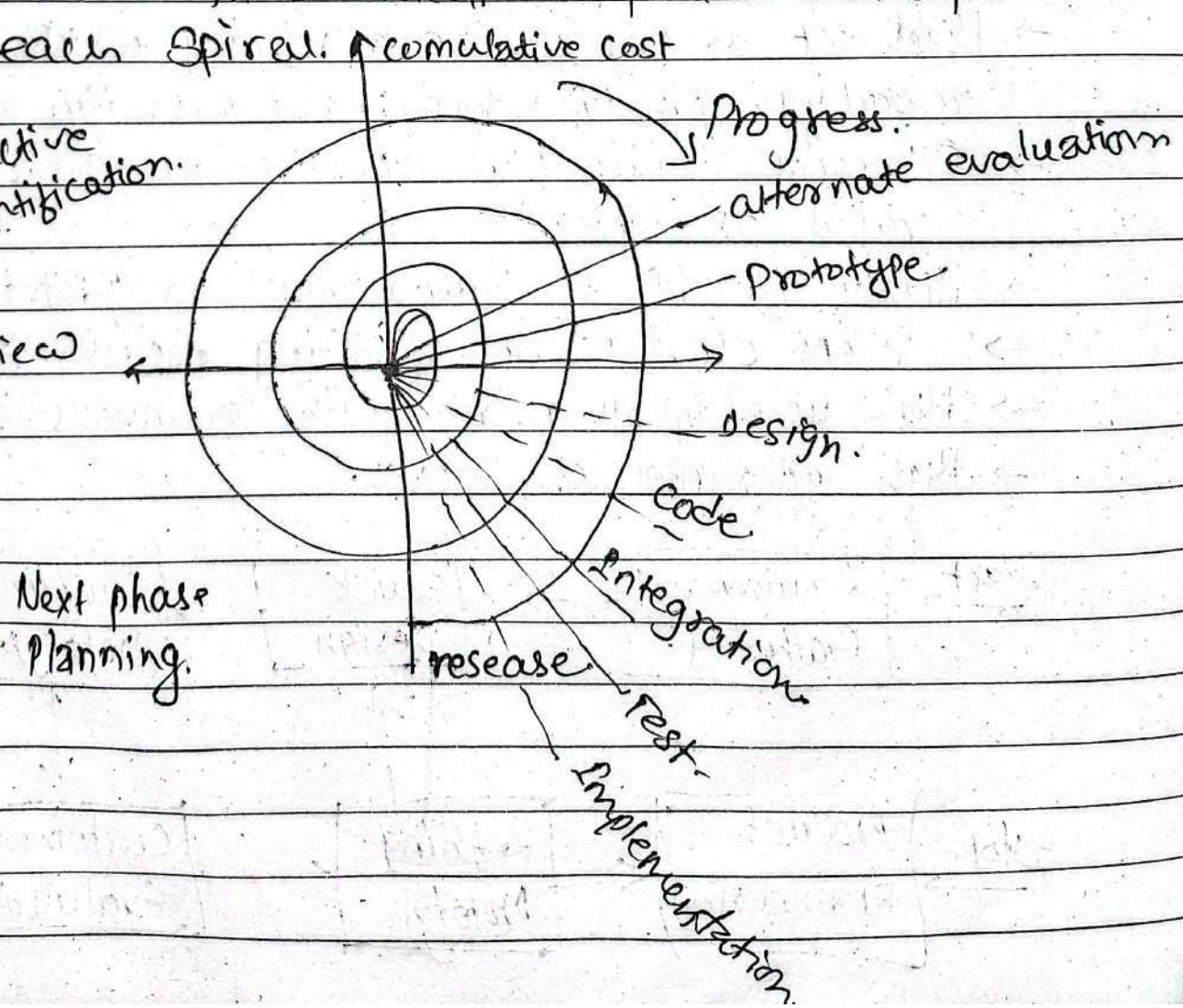
Prototyping Model:

→ Requirements may gets updated frequently as prototype evolves.
→ Requirements are refined ~~after~~ through Multiple prototype
→ High cost as prototypes needs refining which involves reworking. prototype development is costly.
→ Doesnot require high expertise as the focus is on rapid prototyping.
→ Highly flexible to changes based on prototype feedback.
→ Steps overlaps as revisiting occurs.
→ High user involvement with continuous feedback
→ High guarantee of success.



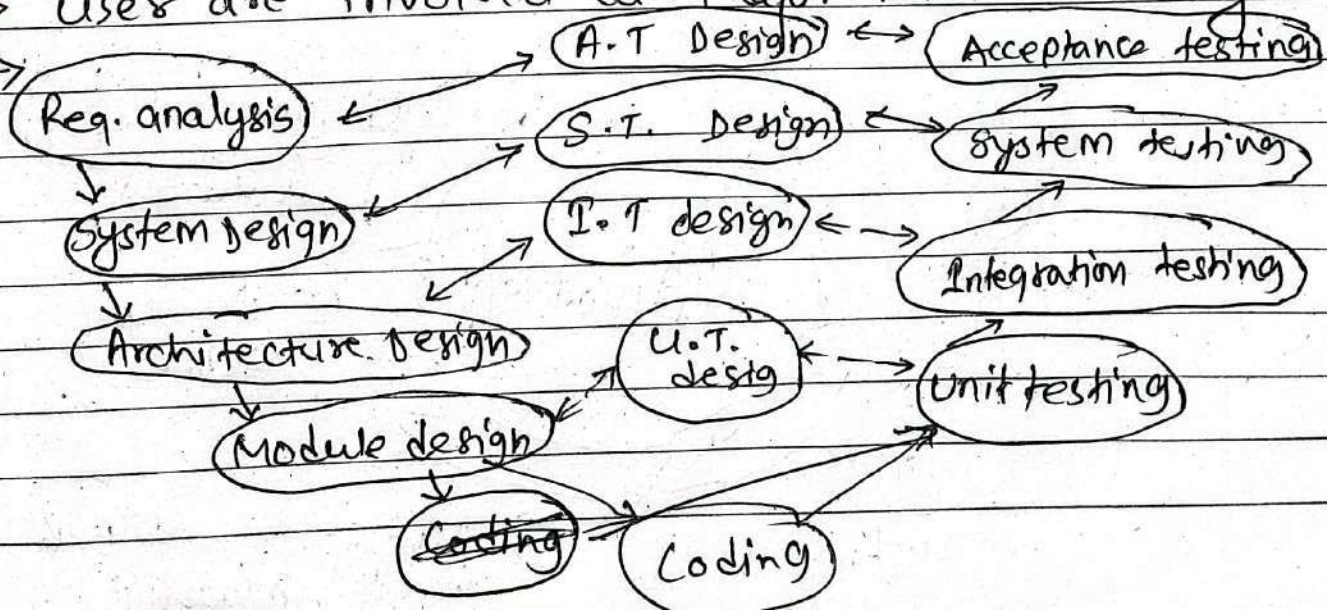
Spiral Model:

→ Requirement Specifies at beginning and refined with each Spiral.
→ Expensive due to iterative nature.
→ High expertise requirement as it involves complex risk analysis & planning.
→ Maintanance is improvemental in each Spiral.
→ High re-usability as components are developed increme
→ High Guarantee of success
→ Documentation required to track the evolution of the Project through Spirals.
→ Phases overlaps.
→ High user involvement as feedback incorporate in each Spiral.



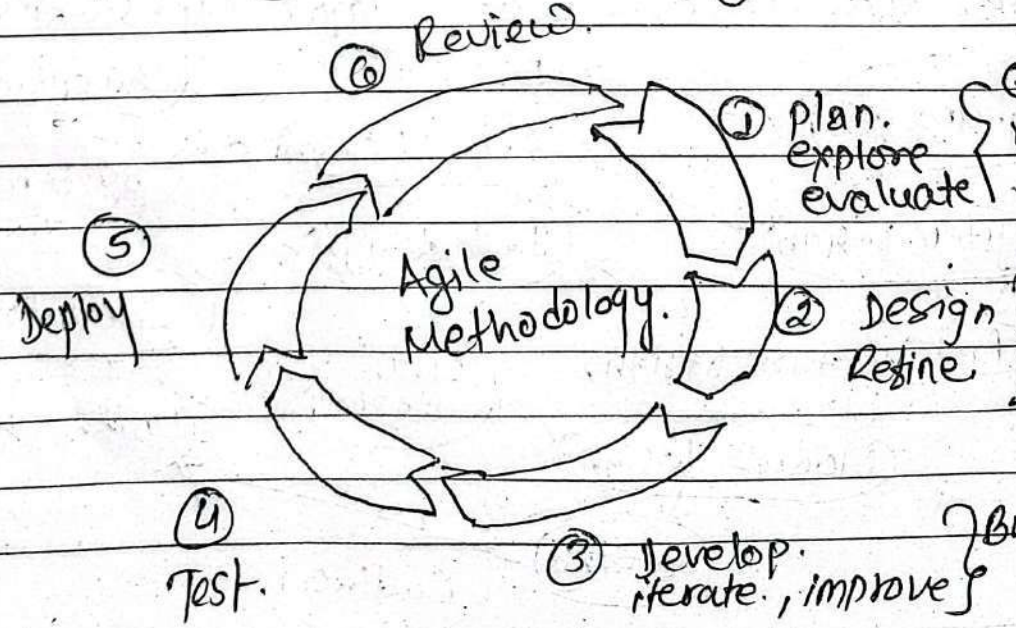
V- Model: → extension of waterfall Model.

→ It is a non-iterative (sequential) process.
→ each phase start after previous one ended.
→ requirements are defined at the start.
→ Costs is predictable as requirements are clear from start.
→ Changes May result in high cost as the structure is inflexible.
→ It is complex due to strict sequential nature.
→ Specific expertise needed in each phase to ensure quality and adherence to preceding Phas
→ Risk are reduced in each phase by rigorous testing.
→ Maintence is High maintance need in early phase and due to rigorous testing, Maintenance become easier later.
→ Detailed documentation is required in each phase
→ User are involved at Major Milestones only.
→



Agile-Model:

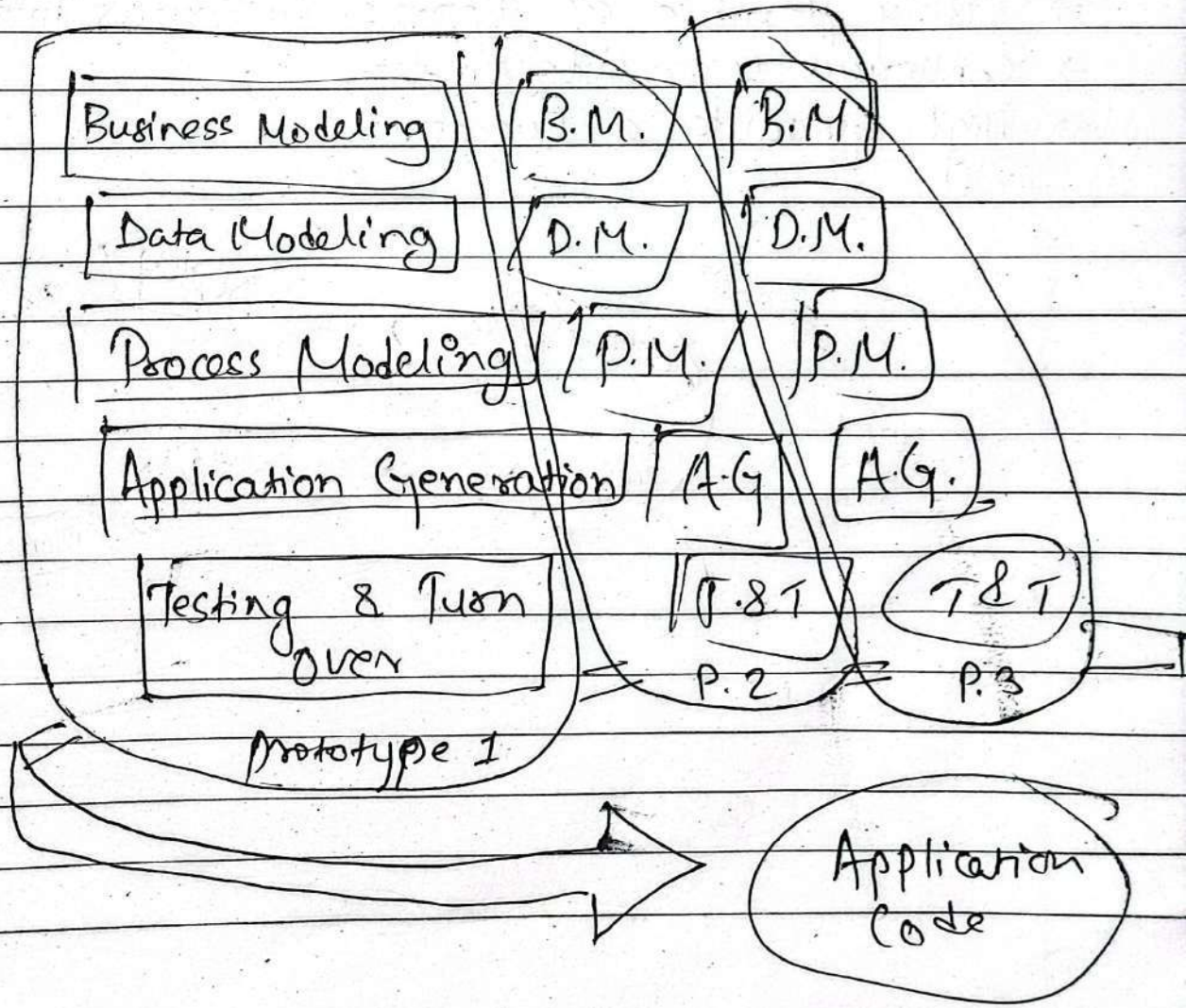
→ It is an iterative & incremental Model that focuses on customer satisfaction by regularly delivering functional software.
→ requirement are supposed to evolve and are refined through constant collaboration with stakeholders.
→ It can be expensive due to regular stakeholder engagement, iterative cycle, frequent changes.
→ Risk are managed easily with regular adjustment of the project and handling unforeseen issues.
→ Development activities may overlap due to iterative work.
→ The modular nature results in reusable code.
→ Less documentation required as the Model focuses on working software.
→ High and continuous user involvement throughout the development process.
→ It adaptability increases the guarantee of success.



Agile Scrum Waterfall, Spiral Model -designing class, defining interface

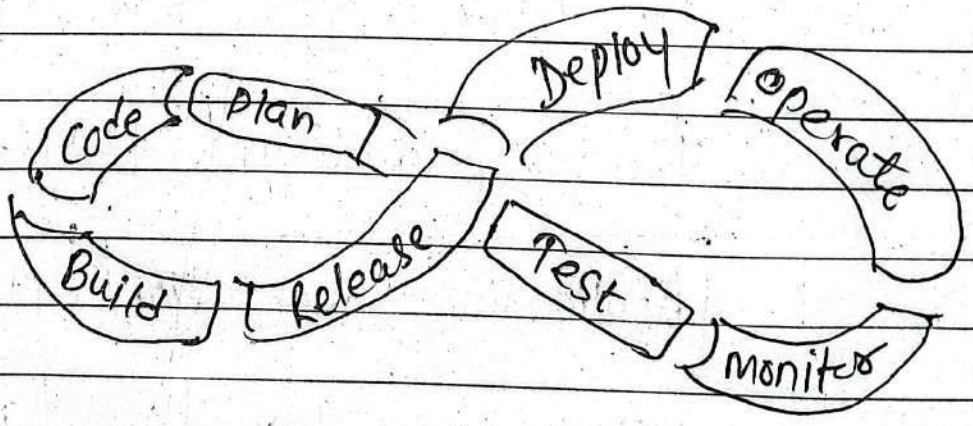
RAD

→ Requirement are refined through iterative user feedback and Prototype adjustment.
→ Cost can be high due to multiple possible iteration and rapid development cycles.
→ It requires high expertise in rapid prototyping & iterative development.
→ Risk are addressed continuously.
→ Maintenance is regular due to regular update
→ only key documents are necessary.
→ High user involvement due to iterative testing.
→ The constant iteration increases the success rate.



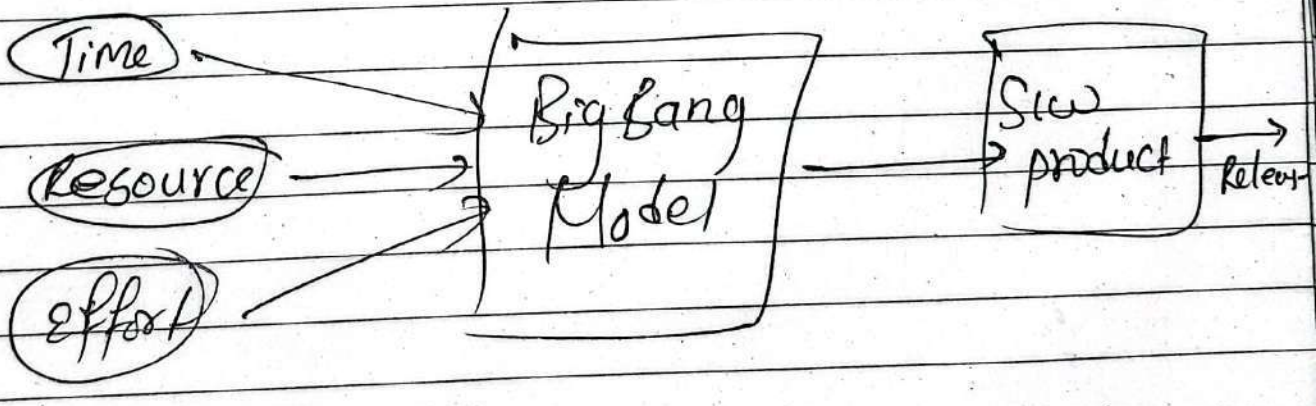
DevOps Model:

→ Requirement evolves continuously due to constant feedback from operations and development.
→ Cost can be high due to substential investment in automation, tooling, cultural change etc.
→ Simplicity is subjective as some process are simple due to automation while other require sophisticated integration.
→ It demands high level of expertise in CI/CD, automat etc. technique.
→ ~~freque~~ Risk involvement is low dee to continuous release and automated deployment.
→ Highly flexible as support Continuous integra & depo
→ Continuous user involvement.
→ High Guarantee of Success.



Bing-Bang:

→ Req. not formally defined
→ development start with general ideas
→ Cost involve is low to unpredictable as there is only little planning.
→ Simple to implement
→ low expertise needed due to lack of formalization
→ High Risk.
→ Highly flexible due to lack of requirement.
→ Minimal to no docs. required.
→ Success rate is uncertain.



Unit-1.

Object Oriented Software Engineering v/s Object Oriented Software Development.

	O.O. Software Engineering	O.O. Software Development.
Scope:	- focuses on entire software development lifecycle, including req. analysis, design, coding, testing, maintenance.	- focuses primarily on writing code using object oriented principles and techniques.
Emphasis:	Systematic approach to S.W development emphasizing methodology, design pattern, and best practices.	- concentrates on implementing the design formed during the engineering phase using OOP.
Activities:	- Includes req. analysis, syst. design, architecture design, implementation, testing, deployment & maintenance	- Primarily involves writing classes, objects, methods, and components using O.O. principles.
Tools & Technique	- utilizes tools and techniques such as UML for design, SOLID Principles, version control, and testing frameworks.	Relies on OOP languages such as Java, C++, Python, etc along with IDE and VCS.
Focus:	- focus on creating scalable, maintainable, robust SW that meet the specified req.	- focus on translating design into working code that implements the desired functionality.
Goal	- to deliver high quality S.W solution that fulfill customer needs & adapt to changing env.	- to implement the design efficiently and accurately a/c to the specification provided.
eg:	- Agile, Scrum, waterfall, Spiral Model	- designing class, defining interface implementing inheritance, polymorphism

Object oriented system Development.

* Object oriented Software development vs. O.O. System development

	O.O. Software Development	O.O. System Development.
focus:	- focuses on developing s.w application using oop.	focuses on developing & designing complex system.
scope:	- Emphasizes the entire SDLC	- focuses on the broader system including hardware and network aspect.
key concept.	- Modularity, code reusability, Maintainability, etc.	- Architecture, security, integration, scalability.
View point	- Microscopic view of classes methods and codes.	- microscopic view of components and system architecture.
Example	- Defining classes like Account, Transactions, etc.	- Designing interaction, data storage, security etc.
Dev. life-cycle	- Req., design, implementation, testing.	- Syst. arch., sw and HW integration, UI designing.
Dependency Mgmt.	- focuses on dependencies bet^n classes and objects.	- focuses on integration between system components.
Maintenance	- Updating individual classes and objects.	- Managing system-wide changes and updates.
Testing:	- Unit testing. Class lvl testing - focuses on code lvl optimization	- Integration, system testing. - Consider HW resource & load balancing.

Object Oriented Software Development.

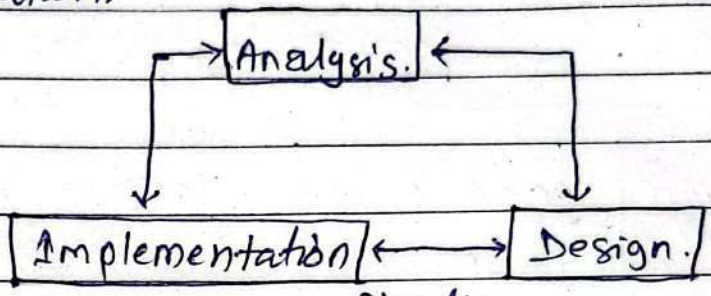
Date
Page

Explain O.O. Sw. Development.

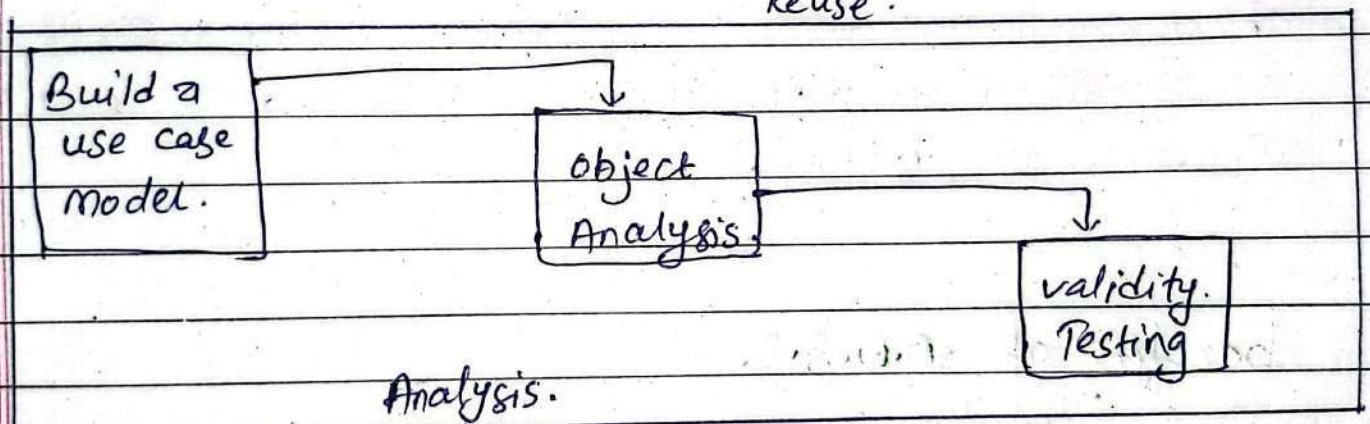
- OOSDevelopment primarily focuses on writing code using O.O principles and technique.

OOSDLC consist of three Macro processes:

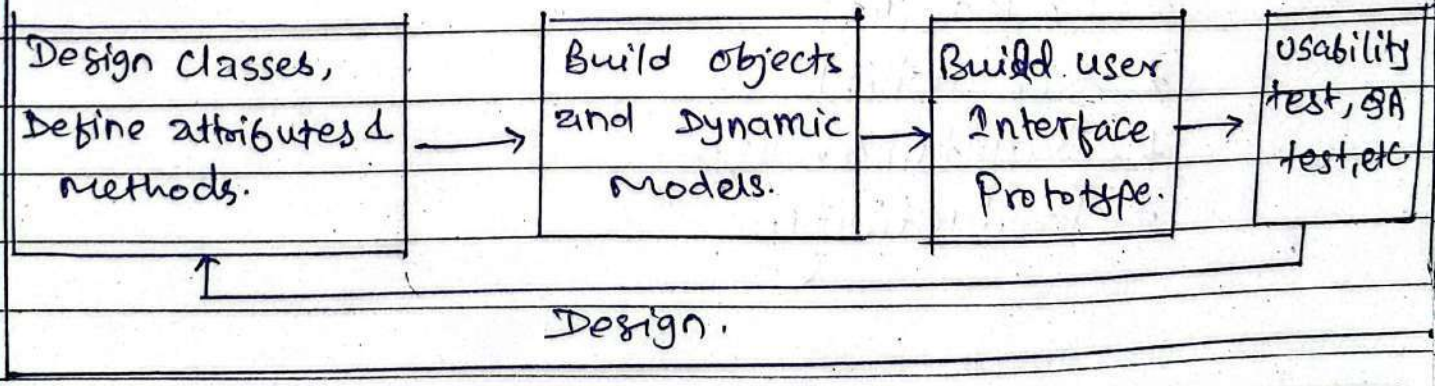
1. O.O analysis.
2. O.O Design.
3. O.O Implementation.



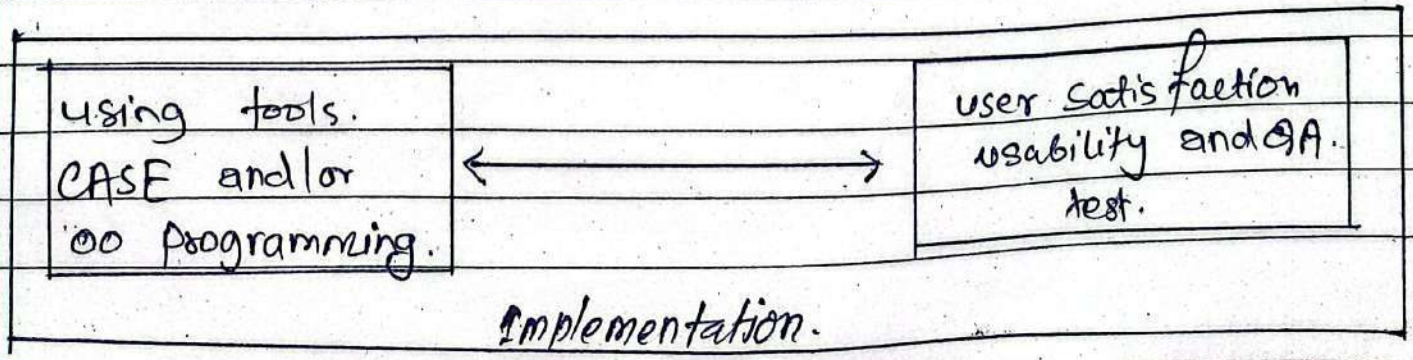
#



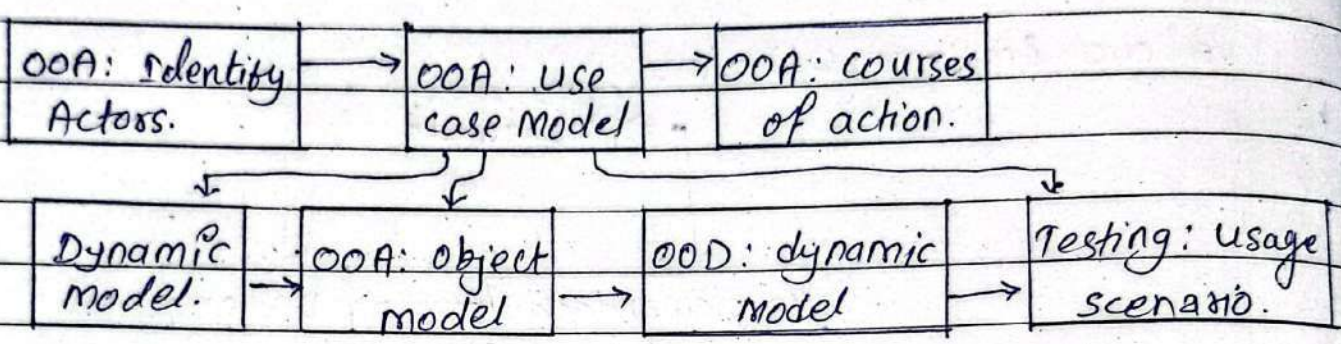
#



#



During,
Object Oriented Software Development, designs are generated which are traceable across requirements, analysis, design, implementation and testing and also it can be directly traced back to user requirements.



Object oriented System Development includes OO Analysis, OO Design, prototyping, component based development, Incremental Testing.

Benefits of O.O.S.D.

* Modularity.
* Reusability.
* Scalability.
* Maintainability.
* Productivity.

Unit-2

	O.O Program Design	O.O System Design.
Scope:	- Designing individual classes within a program	- Designing interaction bet^{n}. software components.
Emphasis:	- Internal logic and organization of class level details.	- Overall architecture, communication of system flow
components	- classes, attributes, methods, and relationships.	- modules, subsystems, data flow and interactions.
Goals:	- Efficient functioning of individual classes.	- scalability, modularity, user requirements.
Example	- Designing classes like Circle, Rectangle,	- Designing interaction between shape editor, drawing canvas
Key concept	- Class attributes, methods, encapsulation, inheritance,	- component interaction, data flow architecture.
View point	- Microscopic view of individual class implementation.	- Macroscopic view of how components work together.
dependency relationship	- Define relationship between classes and methods.	- Define interaction and data exchange between compone
Example	- Book, User, library.	- User interface, Database

Object Oriented Software Development.

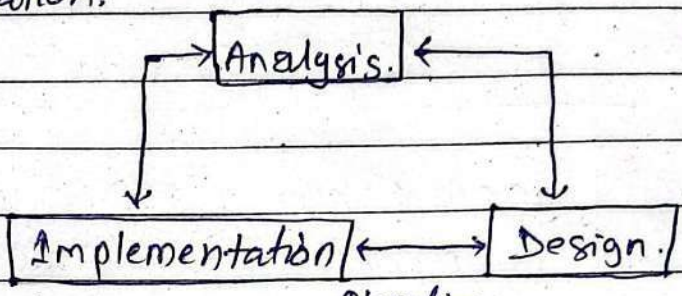
Date
Page

Explain O.O. Sw. Development.

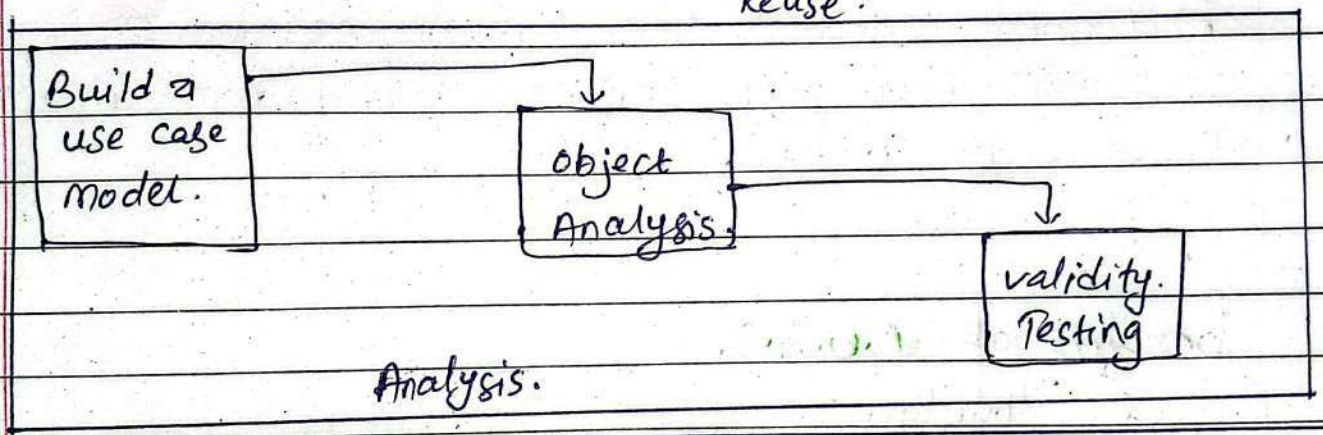
- OOSD development primarily focuses on writing code using O.O principles and technique.

OOSDLC consist of three Macro processes:

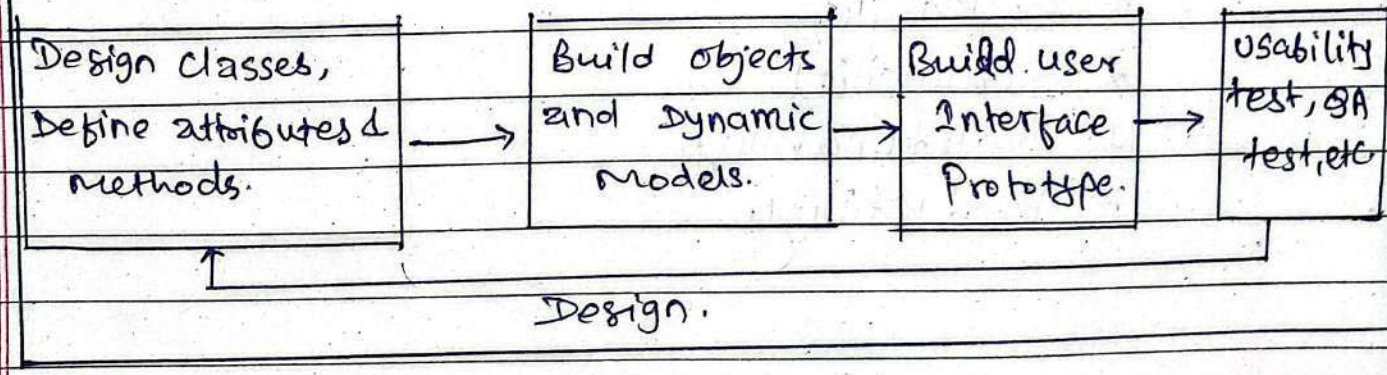
1. O.O. analysis.
2. O.O Design.
3. O.O Implementation.



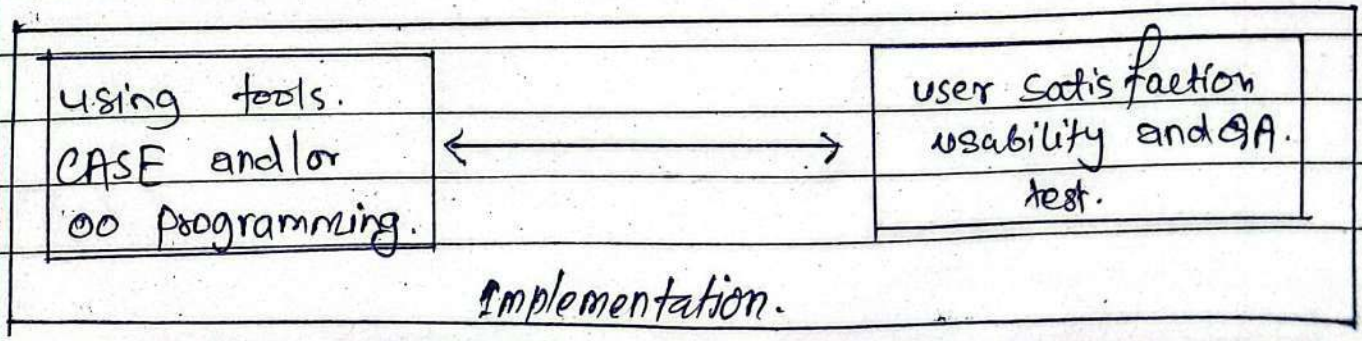
#



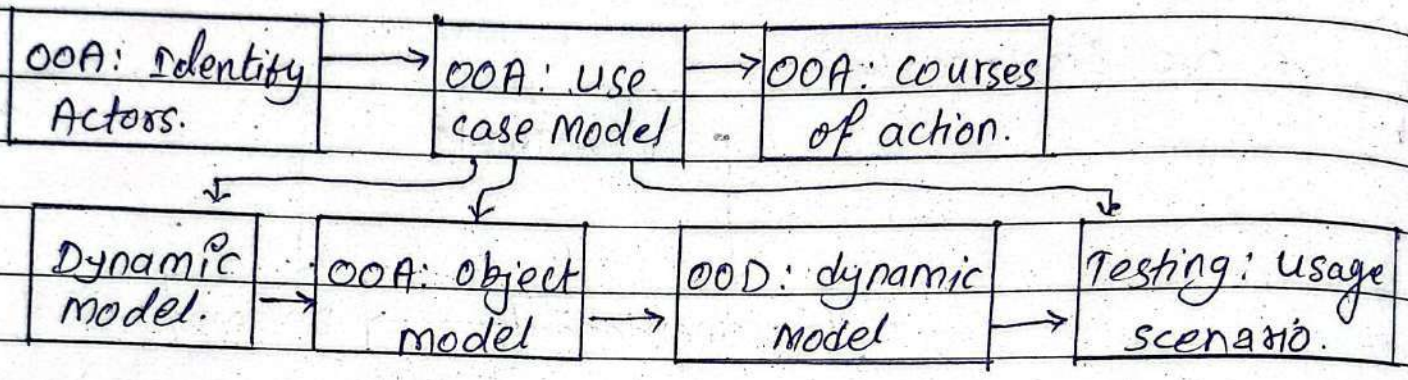
#



#



During,
Object Oriented Software Development, designs are generated which are traceable across requirements, analysis, design, implementation and testing and also it can be directly traced back to user requirements.



Object oriented System Development includes OO Analysis, OO Design, Prototyping, component based development, Incremental Testing.

Benefits of O.O.S.D.

* Modularity.
* Reusability.
* Scalability.
* Maintainability.
* Productivity.

Unit-2

Date
Page

System Development methods.

- function / Data oriented
  - ↓ Requirement SREM / RDD.
  - * Software Engineering methodology.
  - RDD.
  - * SA/SD.
    Structured Analysis / Structured Design.
- object oriented.
  - → OOA alc to COAD.
  - → OOD alc to Booch.
  - → OOSE.

* OOA alc to COAD – a process of identifying the objects and class that will be used in a S.W System and describing their relationsh and interaction.
* COAD → constructive Object Oriented Analysis and Design.
  It is based on following Principles:

- Objects.	- Abstraction.	- Polymorphism.
- classes	- Encapsulation.	
- Relationship	- Inheritance.	

* OOD alc to Booch – a process of defining the structure and behaviour of software aspects.
* Booch define 6 types of diagram than can be used to document OOD process:

- class Diagram.	- state transition Diagram	- module Diagram
- Object Diagram.	- Interaction Diagram.	- process Diagram

* OOD Process alc to Booch involves the following:

- Identify the object.	- Define the behaviour.
- Define the classes.	- Validate the design.
- Define the relationship.	

function/data method vs. object oriented method.

function/data method.	object Oriented method.
- function is active and have behaviour, while data is passive info holder which is affected by the functions.	- function and data is combined as integrated objects.
- system is typically broken down into function and data is passed between those functions.	- System is broken down into objects, function and data will be methods and properties of that objects.
- A system developed using this approach is often difficult to maintain.	- This approach is easier to understand hence system is more maintainable.
- All function must know how the data must be stored. i.e. Data structure.	- Each objects are defined internally that includes defining the info that each object must hold.
- system are often quite unstable as a slight modification can cause major consequences.	- Items with low modification probability are naturally identified and can be isolated at early stage.
- requ. specification is formulated in human language which creates semantic gaps bet the internal views & external views of system.	- objects are naturally occuring entity hence this approach will model the application domain more efficiently reducing. semantic gap.

Object Oriented Programming vs Structure oriented Prog.

feature	OOP	SOP.
Basic concept	Based on object which encapsulate data & Method	Based on function and process that operates on data.
Data handling prog. St.	Data is encapsulated within obj. Organized around obj.	Data is separate from function. organized around f^n and Process.
Modularity	High, as class and objects can be reused and Modified independently.	less modular, more focus on procedure and sequence.
Encapsulation	support encapsulation.	Doesnot support.
Inheritance	~	~
Polymorphism	~	~
Abstraction	~	less support for abstraction.
Flexibility & Maintenance	More flexible and easier to Maintain due to Modularity.	less flexible and harder to maintain as code grows.
Focus:	focus on data & Method toget	focus on function & seq. of execu
Design App.	Bottom up approach.	Top-Down approach.
Reusability	High due to inheritance.	Limited reusability.
Example	C++, Java, ~~Phy~~ python, etc.	C, Pascal, etc.

Unit-3

Date
Page

Component Based Software Engineering:

~~Component Based SE vs Traditional SE~~

→ Software development approach that emphasis the creation and integration of software components as modular, reusable unit. SW are build by assembling pre-existing, independently developed components rather than building from scratch. Components encapsulate specific functionality and can be easily combined to create complex SW application.

Characteristics:

→ Modularity, → Reusability, → Scalability.
→ Inter-operability, → Maintenance, → Rapid development
→ Reduced cost, → Independent, → Extensible.

Example:

1. user Auth Component ⇒ handle login, register
2. payment gateway component → payment processing service
3. product catalog component
4. Shopping cart component
5. Order processing Component

Component:

→ A component is a modular, portable and replacable and reusable set of well defined functionality that encapsulates its implementation and express it as a higher-level interface.

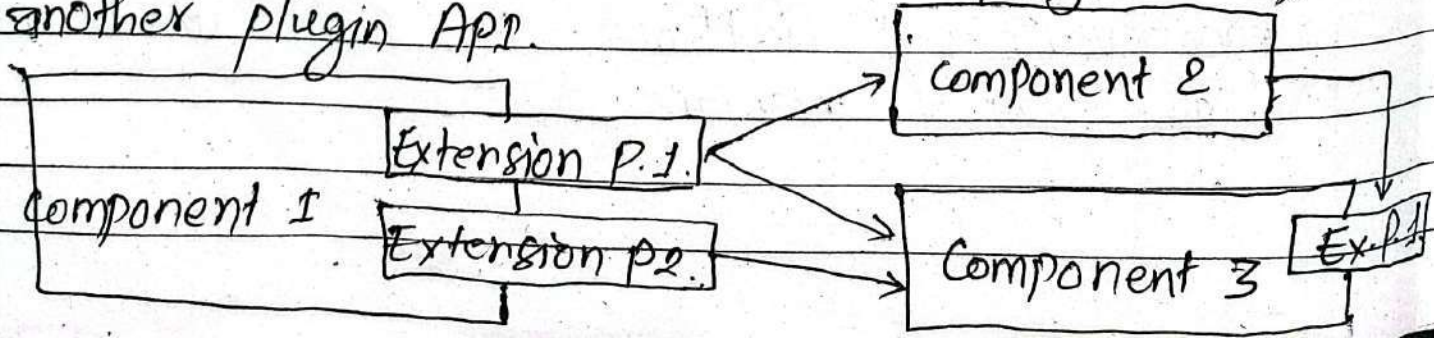
Object Oriented View:
→ Component is one or more cooperating classes, problem domain classes (analysis) and infrastructure classes (design), explained to identify all attributes and operations that are uses during implementation.

Conventional View: functional element or module of a program that integrates the processing logic, internal data structure that are required to implement the processing logic and an interface that helps the component to be invoked & data to be passed

Process related View: Components maintained in library such as UI, grids, buttons, text fields, etc.

Principles:
→ S.S. Is decomposed into reusable, encapsulated Component.
→ Each component has its own interface, define required port
→ A component Should be extended.
→ Component - abstraction, → component interaction

* a Component can extend to other Component and still offer its own extension point. It is the concept of plugin based architecture which allow a plugin to offer another plugin API.



Component Management:

→ development of component is expensive than development of ordinary software. Hence it is used in multiple projects.

→ special component mgmt department is needed which builds component library and maintain it.

following two types of activities are performed for component mgmt:

1. Design of complete component system.
2. Design of individual component.

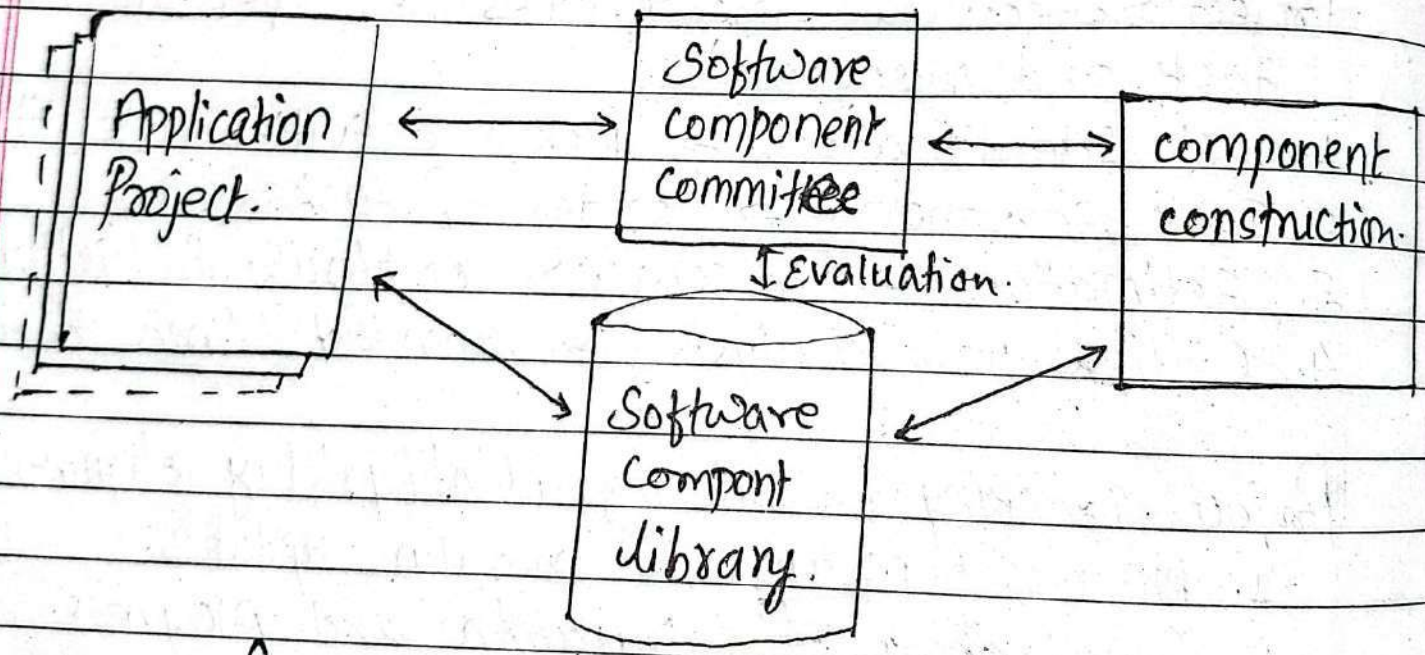


fig: An organization of component mgmt.

Real Time System:

→ A system in which correctness depends not only on the logical result of computation but also on the time at which the result are produced.
eg. flight control system, telephone exchange.

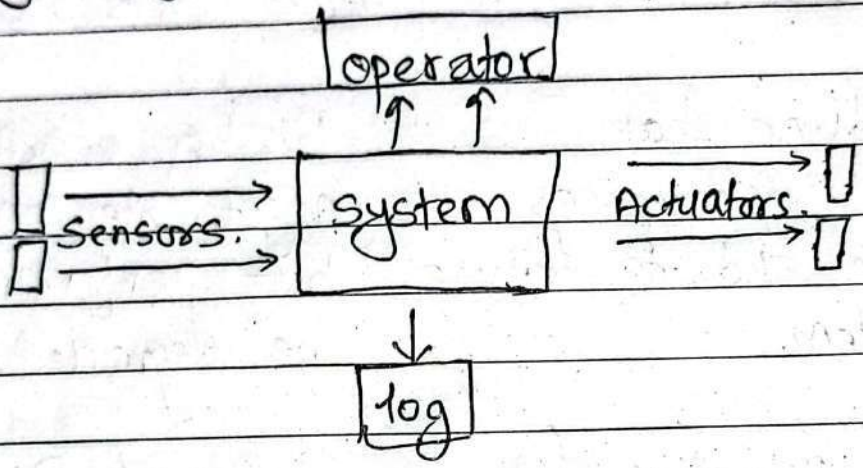


fig: Abstract model of RTS.

→ Sensors and Actuators provides real time view of application behaviour.
→ RTS has means of observing whats going on. controlling, processing from operator and keeping history via logging.

Types of RTS:

1. Hard RTS:
2. Soft RTS:

Hard RTS	Soft RTS.
→ has strict timing which is non-negotiable.	Timing constraint is flexible and negotiable
→ Deadline is critical that means must meet deadline	Deadline miss is tolerable hence can occassionally miss.
→ Missing deadline has catastrophic consequences as well it can lead to failure of the system.	Missing deadline has negotiable consequences but performance may be degraded.
→ High system priorities for real time task.	Balanced Priority and task may be non-rt.
→ Resource allocation are predictable and deterministic	Resource allocation may vary based on load.
→ Complexity may be simple due to strict requirement.	Complex and allows more general purpose use.
→ Aerospace, medical devices, safety critical system.	multimedia streaming, online game streaming
Airbags deployment, ABS brakes ATC, etc.	Video streaming, online chatting.

Issues in RTS:

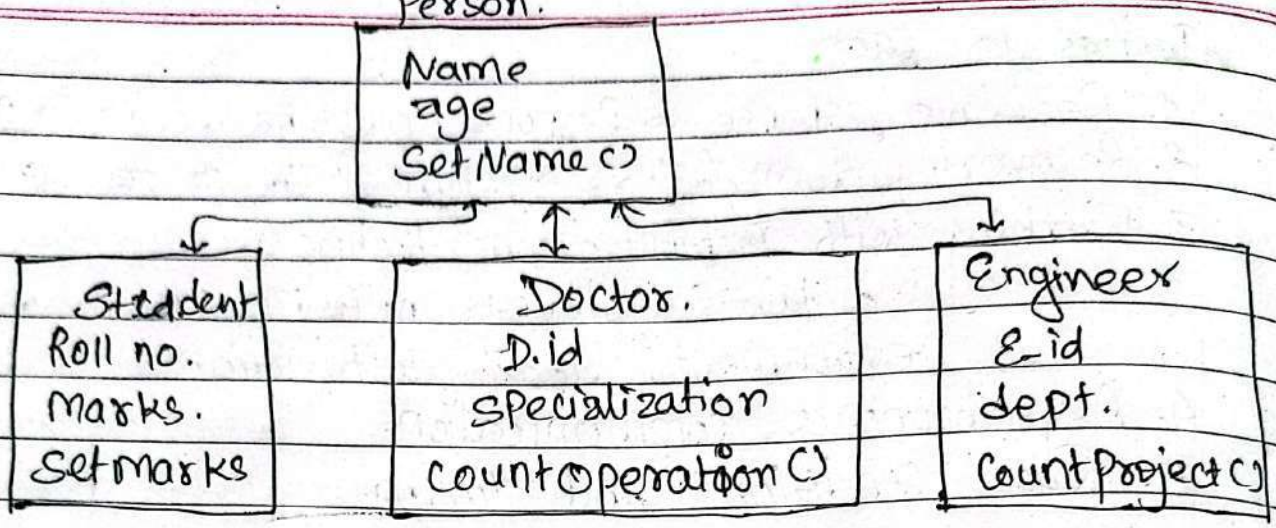
1. Real time response: → Response must be receive in predefined time.
2. Recovering from failure → failure को time में पहिचान reliable service दिनु
3. Working with Distributed Architecture:
   - maintain Consistency, initialize system in inter-process environment, load distribution.
4. Asynchronous communication.
5. Race condition and Timing.

Object Oriented Data Model:

→ To represent complex real world problem there was a need for a data model that is closely related to real world. object oriented Data Model represents the real world problem easily.

In Object oriented data model, data and their relnship are contained in a Single Structure which is referred as object with different attributes. All objects have multiple relationship betⁿ them. Basically it is Combination of OOP and RDBMS as it is clear from below figure.

OODM = OOP + RDBMS.



Basic OO Data Model.

object : abstraction of real world.
encapsulate data and code into single unit
provides abstraction by hiding data.

Attributes
methods.
Class.
Inheritance.

features:
1. Query language.
2. Recovery.
3. ACID transaction
   - Atomicity
   - consistency
   - Isolation
   - Durability.

Pros:
- controls DB redundancy.
- Easily maintain Backup.
- multi-user interface

Cons:
- Cost of Hlw & Slw
- Complexity.
- OODM is under-developed hence not accepted widely.

Description

Date
Page

What are the main reason in Construction Phase?
What is done in Construction phase?

There are three main reasons for having Construction Phase:

I. Analysis Model is not sufficiently formal.
→ To change the source code we must refine the object, which operation we should offer, exactly what should the communication beth different object looks like etc.

II. 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 adopt to the environment in which the system is to be implement.

III. To validate the analysis result:
→ In construction we see the result of analysis, if we discover unclear requirement.

Requirement Model → Construction → Design Model
Analysis model → Implementation Model

Requirement Model:

→ R.M. defines the system boundary and functionality the system should offer.
→ It functions as the contact between developer and the order of the system, specially from developer view of what customer wants.
→ This model should be readable also for non-OOSE ~~practicioner~~ practitioners.
→ This model govern the development of all other model hence it is central one throughout the system.
→ This model will be structured by analysis model, realized by design model, ~~implementation by model and~~ implemented by implementation model and tested by testing model.
→ It consist of three parts:
1. Use-case Model.
2. Problem domain object Model.
3. Interface

Use-case Model:

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

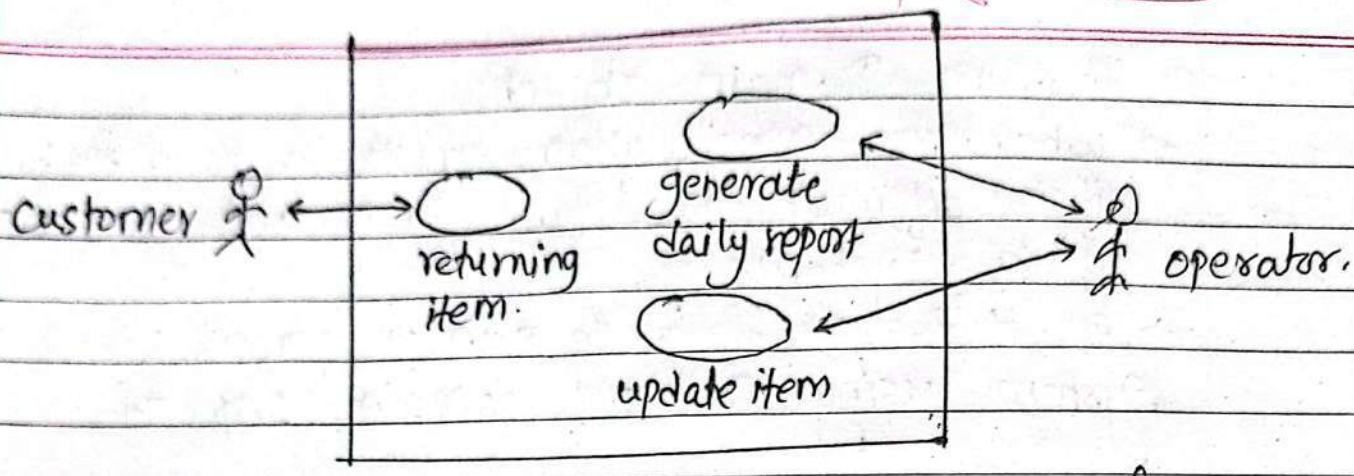


fig: first attempt of the use case model for the recycling machine.

II. Interface Description:

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

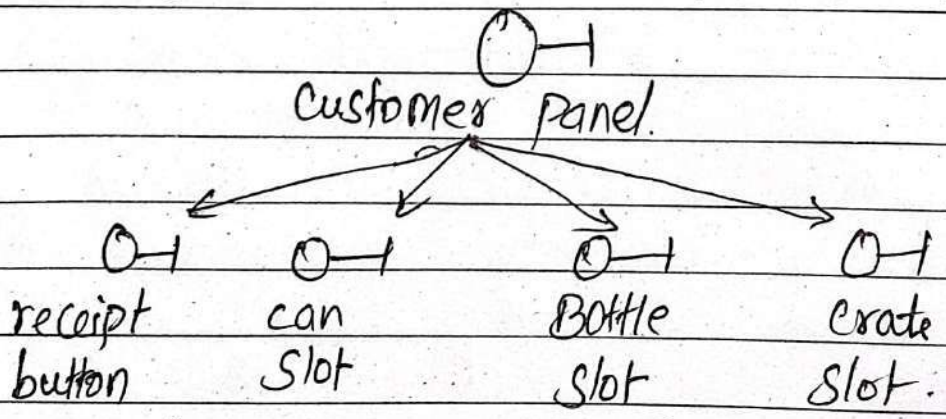


fig: Example of interface design.

III. Problem Domain object:

→ When req. specification exists in vague form, it is difficult to define task and boundary of a system.

→ So, it is good to develop a logical view of the system using problem domain object.

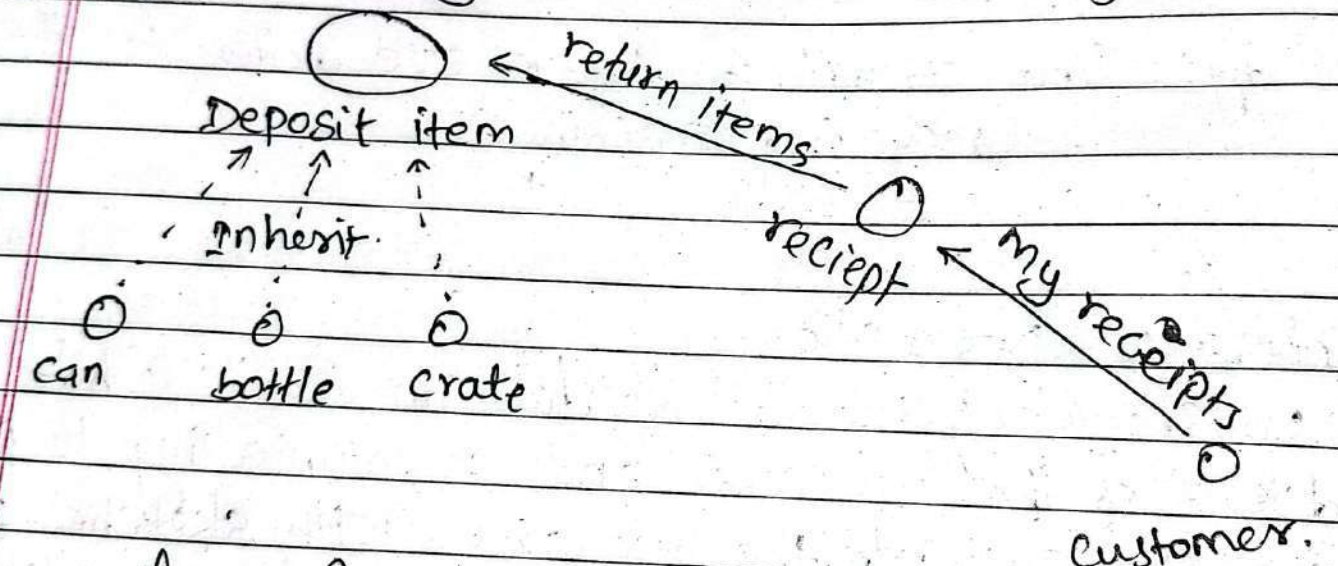


fig: A problem domain Model of the recycling system.

Object Oriented Testing Strategies:

→ Testing is the process of finding differences between the expected behaviour specified by system models and the observed behaviour of the ~~sys~~ implemented system.
→ from modeling pov testing is the attempt to show that the implementation of the system is inconsistent with the system model.
→ The goal is to design test that ~~exercise~~ exercise defects in the system and to reveal problems.
→ It is contrary to all other activity as analysis, design, implementation, etc are constructive activities whereas testing is destructive as it ~~try~~ tries to break the system.

following are the tech. for increasing the reliability of a system:
- fault avoidance :→ try to prevent insertion of fault. (inspection)
- fault detection :→ (debugging) (testing)
- fault tolerance :→ ~~+~~

Types of Testing:

1. unit testing.
2. Integration testing.
3. System testing.
4. Regression testing.
5. Operation testing.
6. full-scale testing.
7. Performance testing.
8. Stress testing.
9. Negative testing.
10. Acceptance testing

Alpha testing.
Beta testing.

Unit-Testing
: Single unit of the system is tested.

Integration testing:
multiple units combinedly tested to check whether they are working fine or not.

System testing:
All the components integrated together and the overall system is tested.

Performance test:
measure the overall performance with varying data load.

Stress test:
→ Extreme limit of the system.
→ Overload test.

Negative test:
- perform to break the system and verify the response during unwanted input.

Acceptance testing: - tested with real data.

Alpha testing — User of the software work with developer team to test the software at developer site.

Beta testing: release of S.W is made available to user to allow them to experiment.

User test

Black Box testing: → testing that examines the functionality of an application without peering into its interal structure. Tester is aware only about what the syste is supposed to do and not how it do that.

Adv: → tester doesn't need knowledge of code
→ unbiased as tester are independent.
→ Helps identifying missing functionality

Disadv → limited by no. of possible input.
→ Not effective for complex logic

White Box testing: Testing that examines the internal structure of an application. (glass box or clear box testing) focus on internal code structure, data & control flow.

Adv: → Thorough as it cover internal st.
→ Helps in code optimization and finding hidden error.
→ useful in verifying loops, condition, etc.

Disadv: → Require detailed knowledge of code.
→ More complex & time consuming.

Integrating together:
→ Req. analysis & design (B.B. testing).
→ Unit testing (White Box testing)
→ Integration (Both)
→ System testing (B.B. Testing).
→ Acceptance testing (B.B.T)

Desc.

Managing Object Oriented Software Engineering

OOSE is known for its high level design capabilities.

Managing OOSE involves following key factors:

* Project selection and preparation.
* Product development organization
* Project organization and management
* Project staffing
* Software quality assurance
* Software metrics.

Project Selection and preparation:

→ Select a real Project that is important and has scope in the market but not with a tight time or any hard constraint.

→ Select a problem domain that is well known & well defined

→ Select People experienced in system development, who have positive view of changes. The Management should have confidence in them.

→ Select Project Managers 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~~.

Selection:

~~Unit: 4~~

~~→ All perso~~

Preparation:

→ All personnel involved in the new order of work need education and training.

→ Give strict method process definition, more emphasis ~~precise~~ can be put on formal education and trainings.

→ A new development process involves a lot of changes which brings potential risk.

→ Those risk can be managed by following below mentioned steps:

1. Risk identification.
2. Risk valuation.
3. Managing the risk.

Preparation

Product development Organization:

→ Product development is to develop different models in sequence

→ The first model to be ~~developmented~~ developed is requirement Model:

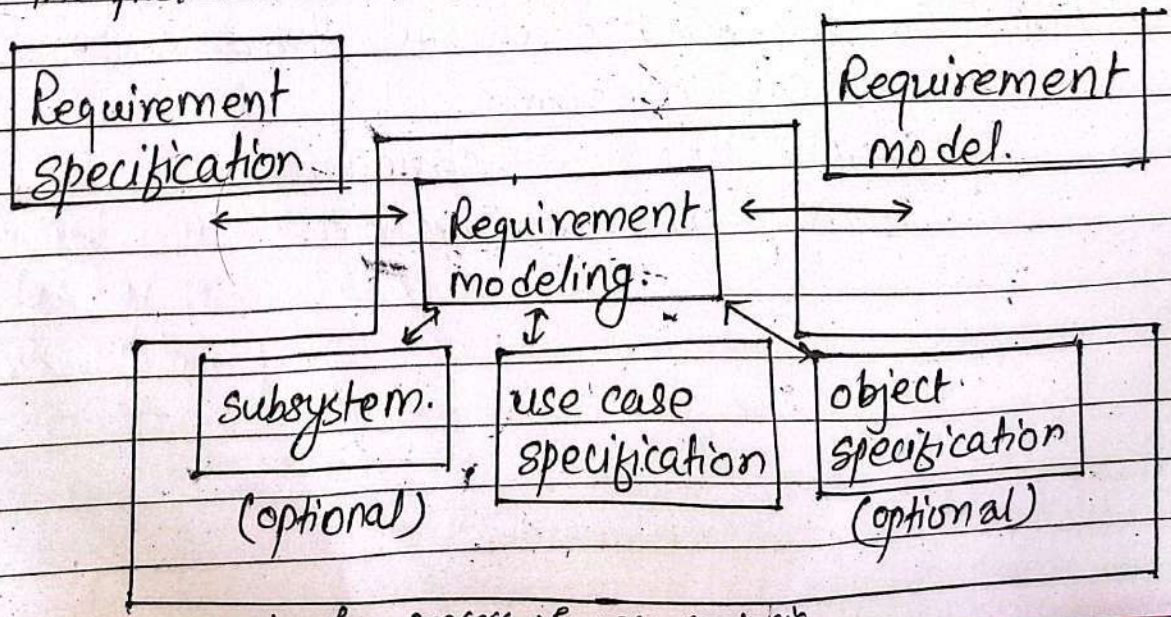


fig: process of req. analysis.

→ Requirement Analysis process delivers a well-defined result: requirement model with use-case specification
→ Next is analysis process which forms the well defined result process model.

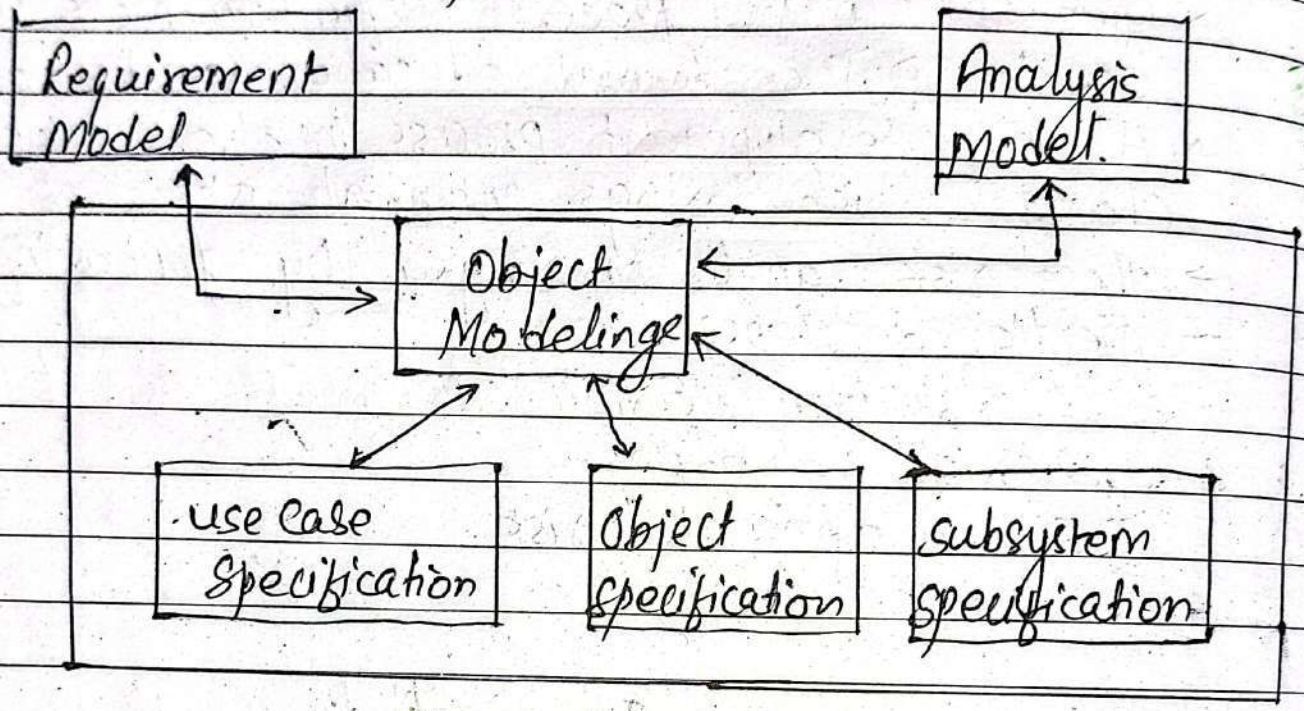


fig: the process of robustness analysis.

→ Analysis model is the input for the Construction process
→ The Construction process has three-sub process
1. use case Design.
2. Book Block construction.
3. Subsystem construction.

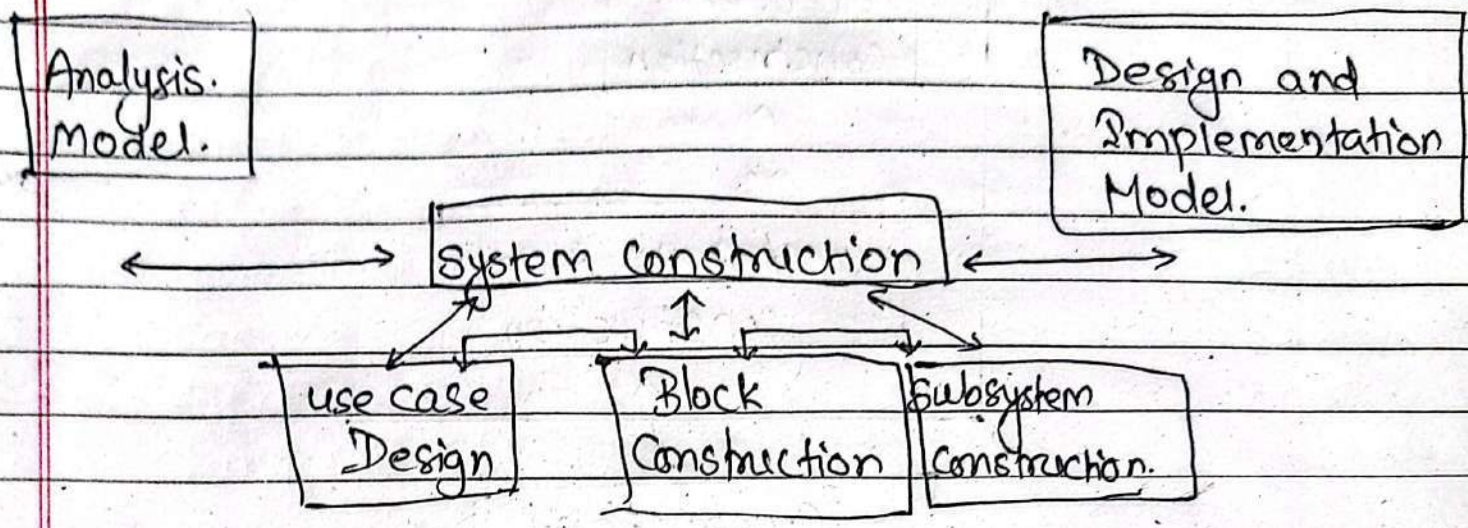


fig: The Process of Construction.

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

→ The next is testing process.

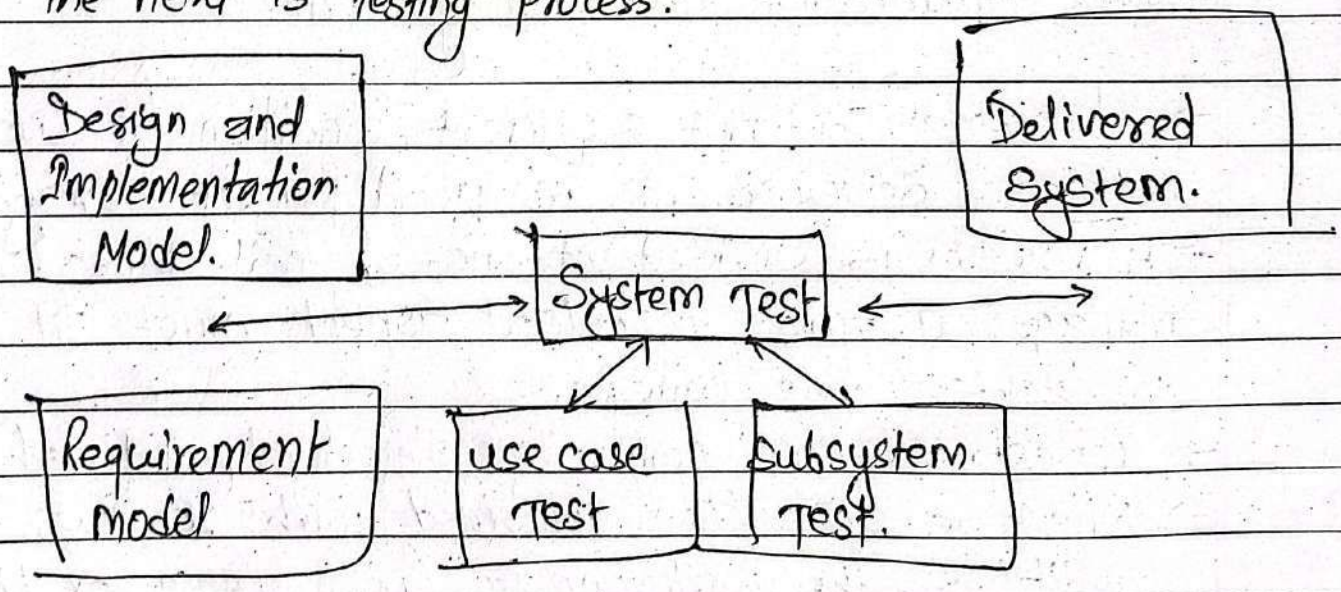


fig: Testing Process.

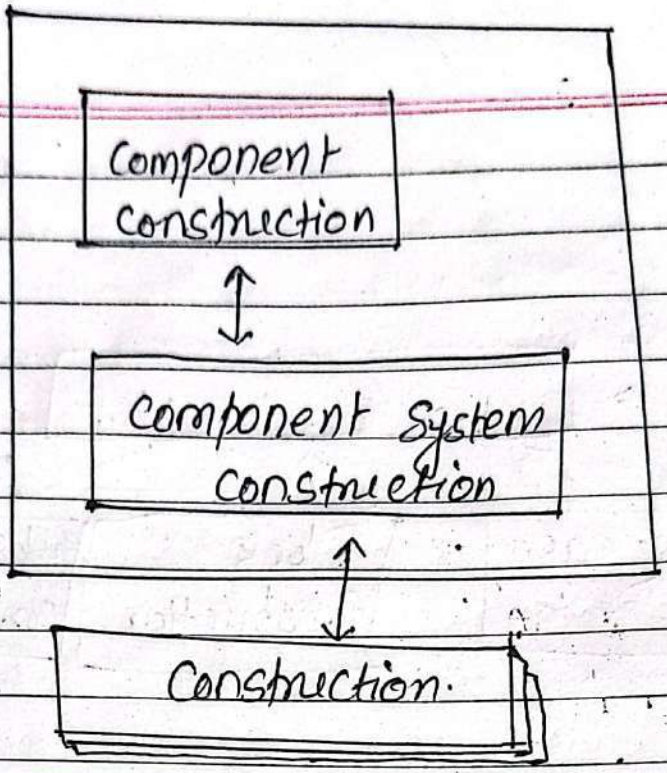


fig: the Component process interacts with several Construction process.

Project Organization and Management.

→ A necessary but not sufficient, condition for Successful Software development is good project Management.
→ A project is divided into no. of milestones and the Managerial and technical aspect must fit together to achieve this milestone.
→ Milestones are concrete, objectively defined variable
→ Milestone are often combined with reviews with audit of the workdone so far. This gives better control of the project

MS → milestone

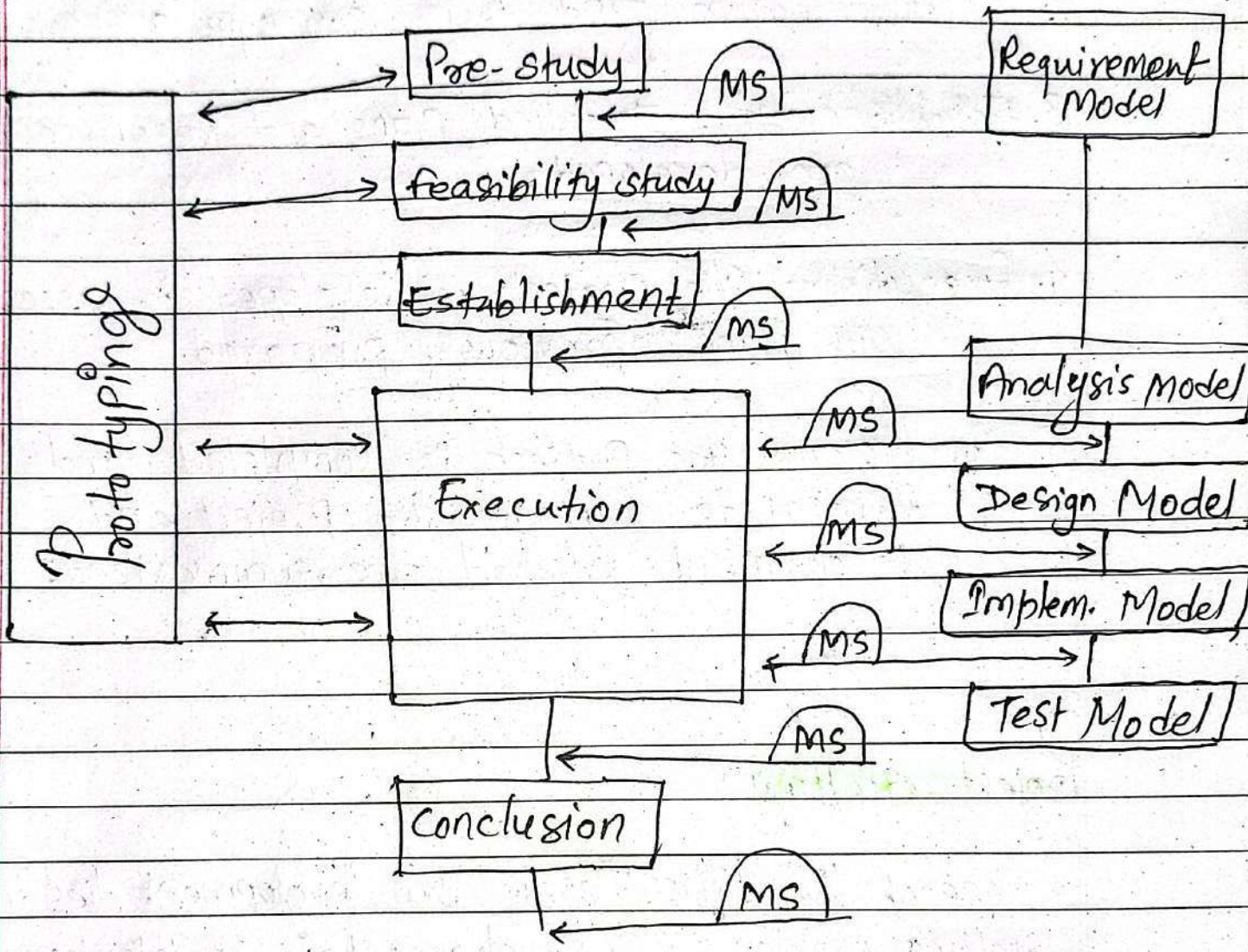


Fig: Managment part of the Project.

Project Management Phases are:

1. Pre-study :- Studies about if the project is practicable or not. It is done by defining and evaluating different kind of requirement to judge project technically and practically.

2. **Feasibility study**: studies different technical alternative and their consequences.

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

4. **Execution**: The project is developed in accordance with plans previously prepared.

5. **Conclusion**: The project is completed and Proposal to improve the project and development method are summarized.

Project Staffing:

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

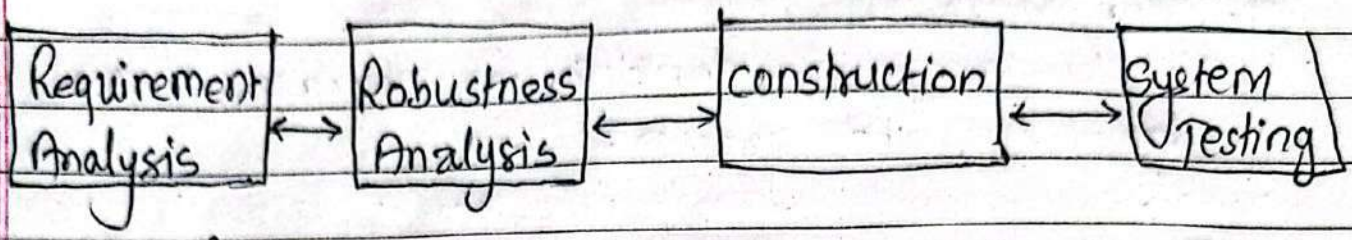


fig: The Co-ordination process in the OOSE.

Different development group can be formed for a Project:

1. System Architecture Group:
   → responsible for making system architecture and delivering coherent idea to the project. They are core of development and same for atleast first three processes. i.e. req. analysis, robustness analysis, and construction.
   → project Manager belong to this group.

2. Requirement analysis Group.
   Initial requirement analysis is done by a small group with interaction with end users.

3. Development Personnel Group:
   more detailed 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 activity. for example: a person specifying a particular use case should also offer use case design and implementation block.

4. Testing group:

Testing is done in a separate phase often by separate group.

Besides above mentioned group, there may exist other roles and groups:

* Methodologist.
* Quality Assurance.
* Documentation, Manuals and training.
* Reuse Co-ordinator.
* Staff.
*

Software Quality Assurance
Software Metrics

} Discussed Earlier.

Unit-4.

Software Quality assurance:

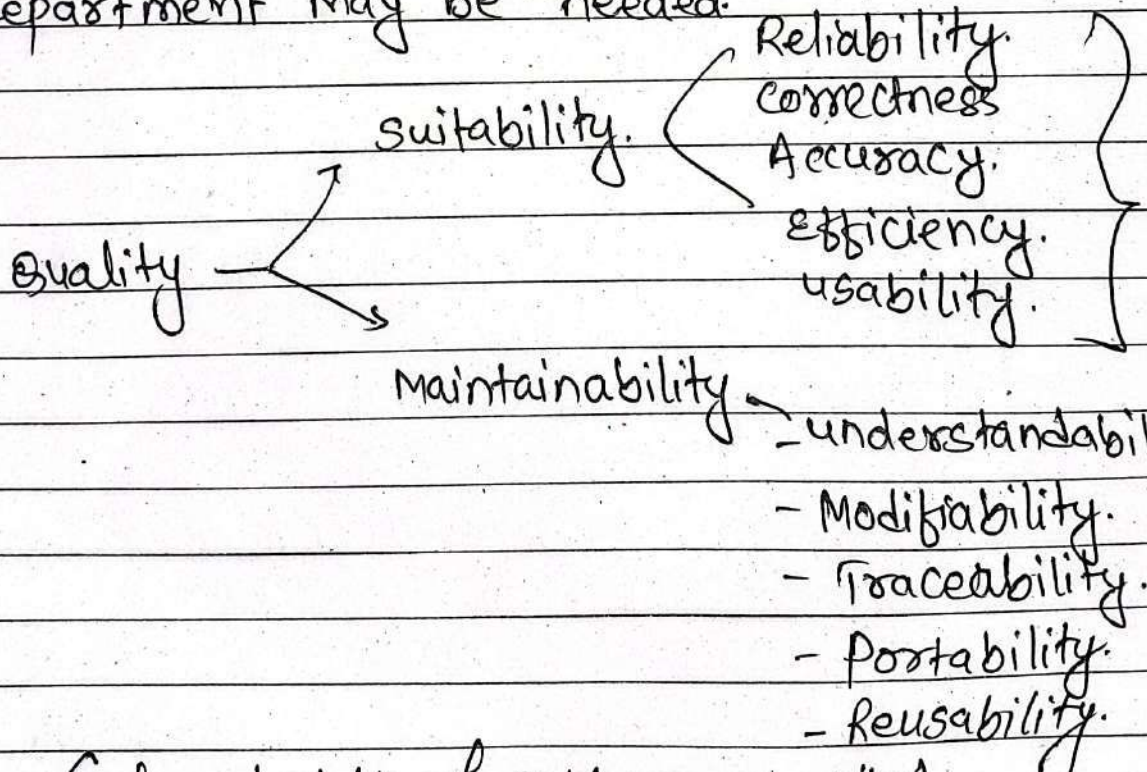
→ It aims that the final product will have an acceptable quality.

→ Cost and time are often tracked in the early stage, rather than quality but quality problem appears later during development.

→ SA focuses on both the project and process.

→ Main tools for quality assurance are:
- development process.
- reviews and audits.
- Testing and Metrics.

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



(characteristic of software quality)

must follow software quality standards like:

- ISO
- IEC
- IEE

Software Metrics: → "If you can't measure it, it's not engineering."

- Software metrics are measure of software characteristics that are quantifiable or countable.
- They are important for many reasons including measuring software performance, planning work items, measuring productivity, etc.
-

following measure are used to describe software metrics.

- Software size metrics
- Software quality metrics.
- Software complexity metrics.

Product Metrics:

1. Software Size Metrics:

This type of metrics measures the size of the SW product or process. Example includes lines of code (LOC) function points (FP), Object points (OP), and use cases (UC).

2. Software quality metrics:

This type of metrics measure the quality attributes of the software product or process. Example includes: reliability, maintainability, Portability, usability, efficiency and security.

3. Software Complexity metrics:

This type of metrics measure the complexity of the software product or process.

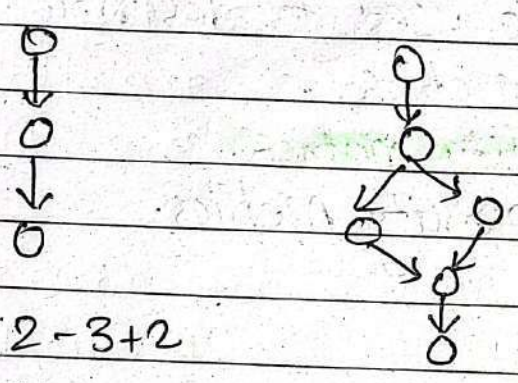
Example includes:

cyclomatic complexity

@ McCabe Complexity metrics.

Mccabe Complexity measures the complexity of graph such that it draws the sequence of program as a graph.

V = connection - nodes + 2.



N = 2 - 3 + 2

= 1

N = 6 - 6 + 2

= 2

fig: McCabe Complexity metrics.

4. Performance Metrics → response time & throughput

5. Time Metrics → lead time, cycle time.

6. Risk Metrics →

* Importance of Software metrics in Sw. development.

Process Metrics

1. Effort Metrics:
   → measure amount of effort required to complete a project.
2. Time Metrics:
   → Total time from beginning to end of processes.
3. Productivity Metrics:
   → amount of work a team complete. (velocity)
   → No. of task completed per unit time (Throughput)

Project Metrics:

1. Cost Metrics:
   → Budget variance - Predicted - actual
   → Cost performance Index -

2. Schedule Metrics:
   → Schedule variance: - planned - actual schedule
   → Schedule Performance Index -

Maintenance Metrics:

1. Change Metrics
2. Stability Metrics:

Imp. of S/w metrics in S/w development.

Software metrics plays a crucial role in the Software development Process for several reason:

1. **Quality assurance:**
   - → Defect tracking: metrics helps in identifying, tracking and managing defects throughout SDLC.
   - → Improvement identification: By analyzing metrics the development tean can pinpoint areas of improvem

2. **Project Management:**
   - → Progress Monitoring: - Metrics provide quantitative data to monitor progress, helping Manager to ensure that projects stays on track.
   - → Risk Management :- Early detection of risk would be easy.

3. **Performance Evaluation:**
   - → Team performance :- Metrics like velocity, throughput, etc provides insights into team performance.
   - → Process Efficiency :- Evaluating metrics related to cycle time, lead time and effort help in assessing the efficiency of development process.

4. **Decision Making:**
   - → Informed Decision :- Metrics offer data-driven foundation for making decision about process adjustment, resource allocation, etc.
   - → Cost Management :- Cost related Metrics helps in budgeting, forecasting, and Manging expense effectively.

5. Customer Satisfaction:
→ user feedback :- Net promoter score (NPS), Customer Satisfaction Index (CSI)
→ performance Metrics :- response time, throughput.

6. Compliance and Standards:
→ Regulatory compliance :- metrics helps in ensuring that S.W development process adhere to industry standards.
→ process improvement Model :-

7. Continuous Improvement:
→ Benchmarking :-
→ feedback loop :-

8. Resource Optimization:
→ Efficient use of Resources:

~~Unit-1.~~

Unit-5:

Object Modeling Technique (OMT) in details:

⇒ OMT is an object oriented analysis, design and implementation methodology focuses on creating a model of objects from the real world and use that model to develop object-oriented software.

→ develop by James Rumbaugh in around 1991.

→ OMT is fast and approach for identifying and modeling all the object making up a system.

→ Details such as class, object, attributes, methods, inheritance, etc. can also be expressed easily.

OMT uses three kind of Models:

- Object Model :- describe the objects in the system and their relationship.
- Dynamic Model :- describe the interaction among objects in the system.
- functional Model :- describe the data transformation of the system.

Object Model : - object, - class, - Attribute, - Operation, - Method, - link, - Association, - multiplicity, - Aggregation, - Generalization, - Inheritance, - module, - sheet, - metadata, - instantiation, - homo morphism.

Dynamic model : - Events, - states, - state diagrams.

functional Model : - Action, - Activity, - Actor, - client, - constraints, - control flow, DFD, - data, - operation,

OMT consist of four phases ; which can be performed iteratively :

1. Analysis: This phase involves the preparation of precise and correct model for the real world problems. performed by analyst. showing important properties

2. System Design: This phase determines all system architecture, concurrent tables and data storage. In simple word, high level architecture of the system is designed by the designer.

3. Object Design: In this phase the object designer build a design model containing data structure and algorithm for objects and class.

4. Implementation: In this phase, Prepared design is converted into fully developed software.

OMT Analysis - consist of iterating following steps:

- generating a problem statement.
- building an object Model.
- building an dynamic model.
- building an functional Model.
- finding operations.

Responsibility Driven Design (RDD):

RDD is an object oriented design technique that focuses on the responsibility of objects. It is a way of thinking about object oriented design that emphasis the behaviour of object rather than their data.

There are following four steps in RDD:

1. Identifying Responsibilities:
   first step is to identify relation of each objects whic
   can be done by asking question like:
   * what does this object do?
   * What actions the object must perform?
   * What information the object needs to store

2. Assigning Responsibilities:
   can be done by analysing:
   * Objects expertise.
   * Objects access to information.
   * Objects relationship with other objects.

3. encapsulating responsibilities:
   Once the responsibilities are assigned, then those responisibilities needs to be encapsulated in the objec so that thes responsibilities should be hidden from the rest of the System which improve modularity and maintainability.

4. Collaborating responsibility:
each object should collaborate with each other to fulfill the responsibility assigned to them. So, those objects must communicate and share their information.

RDD vs. OMT:

	Responsibility Driven Design	Object Modeling Technique.
Focus: 1.	Primarily focus on identifying and allocating responsibility to objects.	① focuses on creating a visual model of the system capturing its structure and relationship.
2.	Creates well-defined methods and collaboration between objects.	② Provides a set of graphical notation to represent object, class and association.
Emphasis: 3.	Emphasis the behavioral aspect of objects, concentrating on what the objects do & how they collaborate to achive the system goal	③ emphasis both structural & behavioral aspects of object. It covers representation of object & class as well as their interactions.
Design process. 4.	Involves identifying responsibilities, assigning responsibilities, Encapsulating responsibilities Collaborating responsibilities between Objects.	④ Involves creating models like class diagrams, State transition diagram and interaction diagram.

	RDD	OMT.
Documentation	⑤ May not rely heavily on formal documentation rather focuses on capturing responsibility.	⑤ It places strong emphasis on documentation through diagrams that represents various aspects of the system.
Application scope	⑥ Detailed level of design focuses on designing methods, responsibilities, and collaboration of individual objects.	⑥ Higher level of design, where focuses on visualizing the overall system structure, classes objects & their relationship.
Notations:	⑦ doesn't have specific graphical notations, often involves textual description and/or pseudo-code to define responsibility and methods.	⑦ Provide set of graphical notation such as class diagram state transition diagram and object interaction diagram.

What is Object Oriented Analysis? Explain with eg.

* It is the process of identifying, understanding and Modeling of the objects and their interaction in the system. ~~The goal~~
* The goal of OOA is - clear and comprehensive understanding of problem domain and its requirement.
* Analysis phase will analyze, specify and define the system which is to be built
* Two different models are developed
  - Requirement model.
  - Analysis model.

Main Purpose:
* How to characterize a system.
* To know what are different relevant Object.
* How do they relate to each other.
* How to specify or model a problem to create effective design.
* Examine requirement; Analyse their implication.

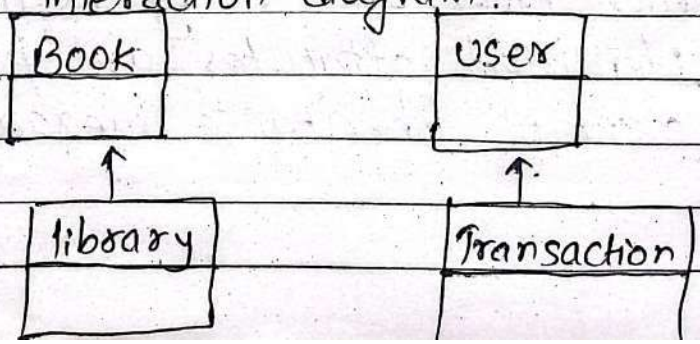
Example:
Problem statement:-
Design a library management system in which user can search, borrow, return books.

1. Identify objects:
   * Users: That uses the system with attributes.
   * Books: represents books with attributes.
   * library: represents library with attributes.
   * Transaction: represents borrow, return.

2. Define Relationships:
   * a user can borrow multiple books.
   * a Book can be borrowed by multiple users.
   * library manages the inventory of books.
   * transaction involves user borrowing, returning. books.

3. Define attributes and methods:
   * Book:
     Attributes: ISBN, author, title, etc.
     methods:
   * User:
     Attributes: Name, contact, email, address, etc.
     method:
   *

4. Create object interaction diagram:



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)
- OOA according to COAD is a process of identifying the objects and classes that will be used in a system and describing their relationship and interaction.
- This Method consist of 5 steps:

1. finding classes and objects:
   - It specifies how classes and objects should found within a problem domain and within content of system's responsibilities.

2. Identifying structure:
   It is done in two different ways
   - generalization: Specializing general spec
   - Whole Part Structure.

3. Defining subjects.

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

3. Defining subjects:
- It is done by partitioning the class and objects model into larges units.
- Subjects are group of class and objects.

5. Defining services:
- It means defining operation of classes.

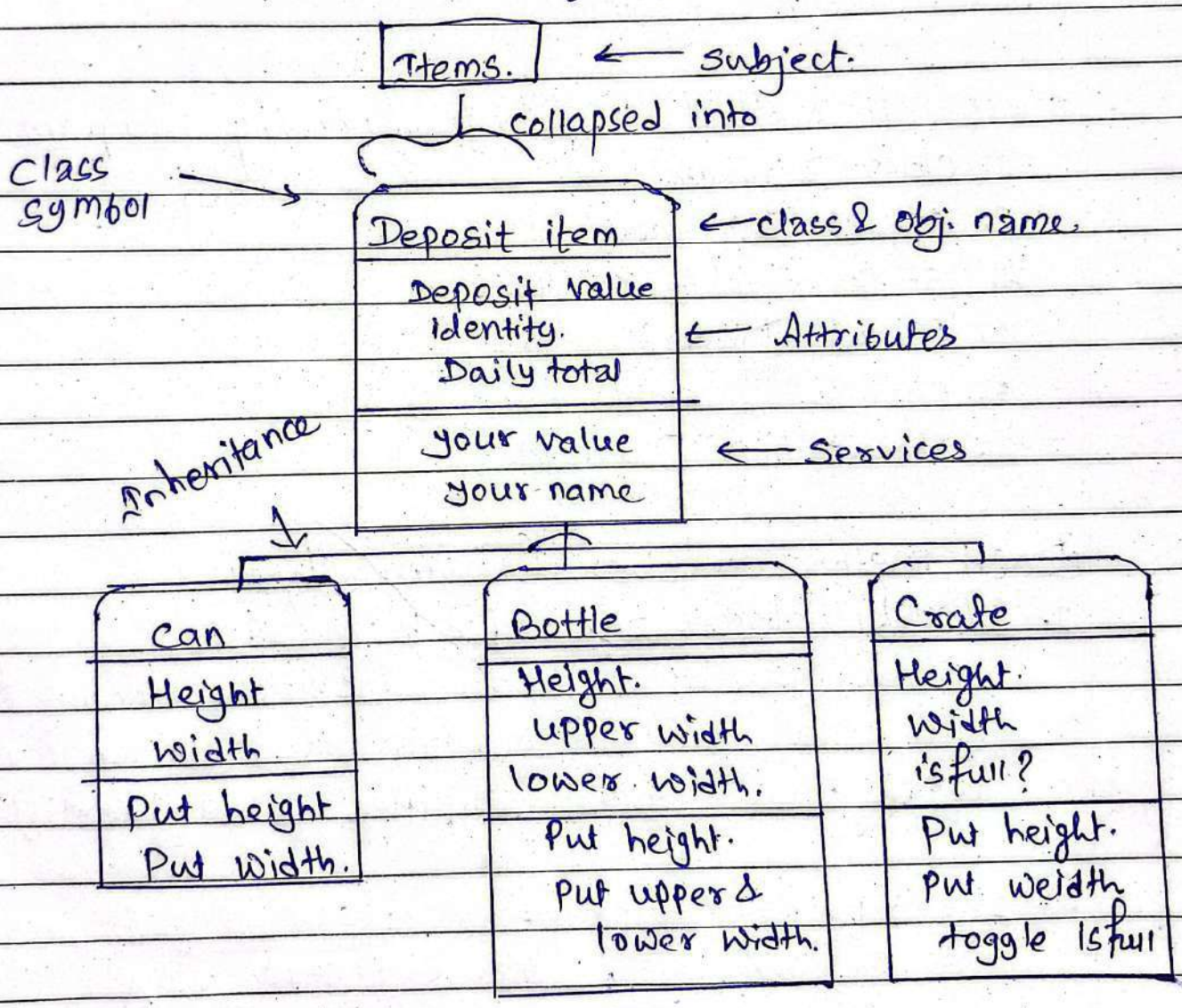


fig: OOA for a part of recycling system.

Object Oriented Design / Booch (OOD/Booch)

- OOD according to Booch is a process of defining structure and behavior of soft objects.
- It is widely used object oriented method that helps us design our system using the object paradigm.
- It covers the analysis and design phase of developing object oriented system.

- The booch method consist of following diagram:
  1. Class Diagram.
  2. Object Diagram
  3. State transition Diagram.
  4. Module Diagram.
  5. Process Diagram.
  6. Interaction Diagram.

Booch method Involves following steps.

Identify class and objects:
- involves finding key abstraction in problem space and important mechanism that offer the dynamic behaviour over several objects.

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

Identifying class and objects relationship:
- involves extending the previous activities to include the relationship between classes and objects and to identify how these interact with each other.

4. Implementing class and objects:
- involves delivering into class and objects and determining of how to implement them.
- A decision is made on how to use particular programming language to implement this ~~usage~~ classes.

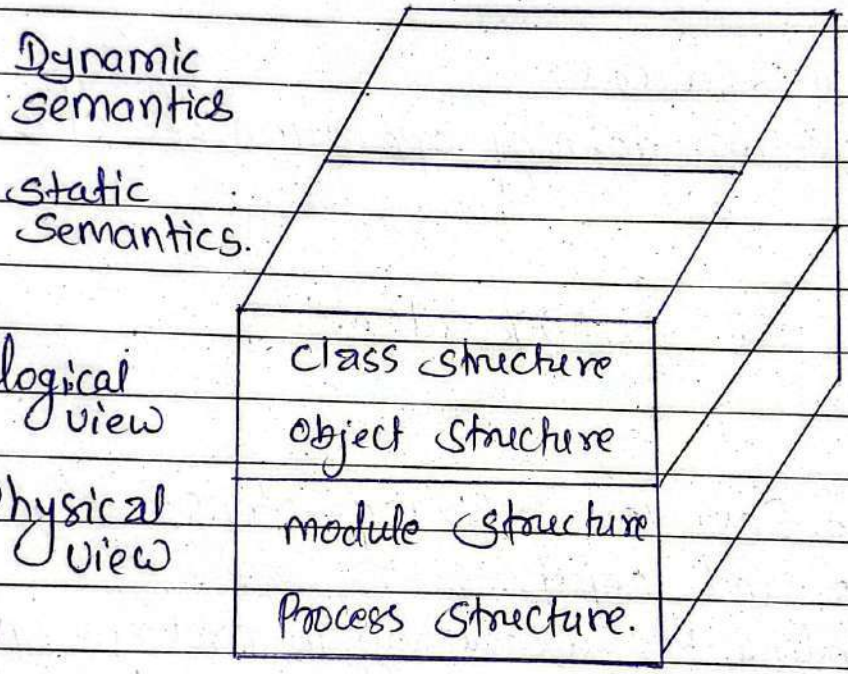


figure: Documentation aspect in OOD.

Booch consist of macro & micro development process.

1. macro development process: It serve as controlling framework for micro process and its main concern is technical management of system. It consist of following steps:
   - (i) conceptualization.
   - (ii) Analysis and Development of model.
   - (iii) Design or Create system Architecture.
   - (iv) Evolution or Implementation.
   - (v) Maintenance.

- HOOD is a ... software units, based on identification ... Problem domain

Date
Page

2. Micro Development Process:

It is description of day to day activities | operation by Software developer.

It consist of following steps:

(i) Identify objects and classes.
(ii) Identify object and classes semantics.
(iii) Identify object and classes relationship.
(iv) Implementing classes and objects.

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 start after analysis and extends down to coding and testing.
- It has object oriented Paradigm.
    1. Unit of decomposition is no more than action, but the Object.
    2. An object is an abstraction of a real world entities
- The hierarchy described in HOOD takes two phases:
    1. Uses :- dependency of one object on another service.
    2. functional Decomposition:- objects split into child object to give functionality of parent object

HOOD has four phases:

1. Problem definition:
    - A statement of the problem is made to provide a content for the current object level.
2. Development of informal solution strategy:
    - solution to the Problem defined previously is outlined
3. formalization of the strategy:
    - major concept of the informal solution strategy is extracted to formalize the Soln.

4. Formalize the Solution:

- It is done by developing a formal model of each identified objects.
- This is performed in five steps:

a. Identification of objects: It is done by extracting the nouns from the informal solution strategy and selecting appropriate one.

b. Identification of operation: is done by extracting the verbs from the informal solution strategy.

c. Grouping objects and operations: involving attaching each operation to an appropriate object.

d. Graphical description is done by using HOOD graphical formalism.

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

(P.T.O)

HOOD design:

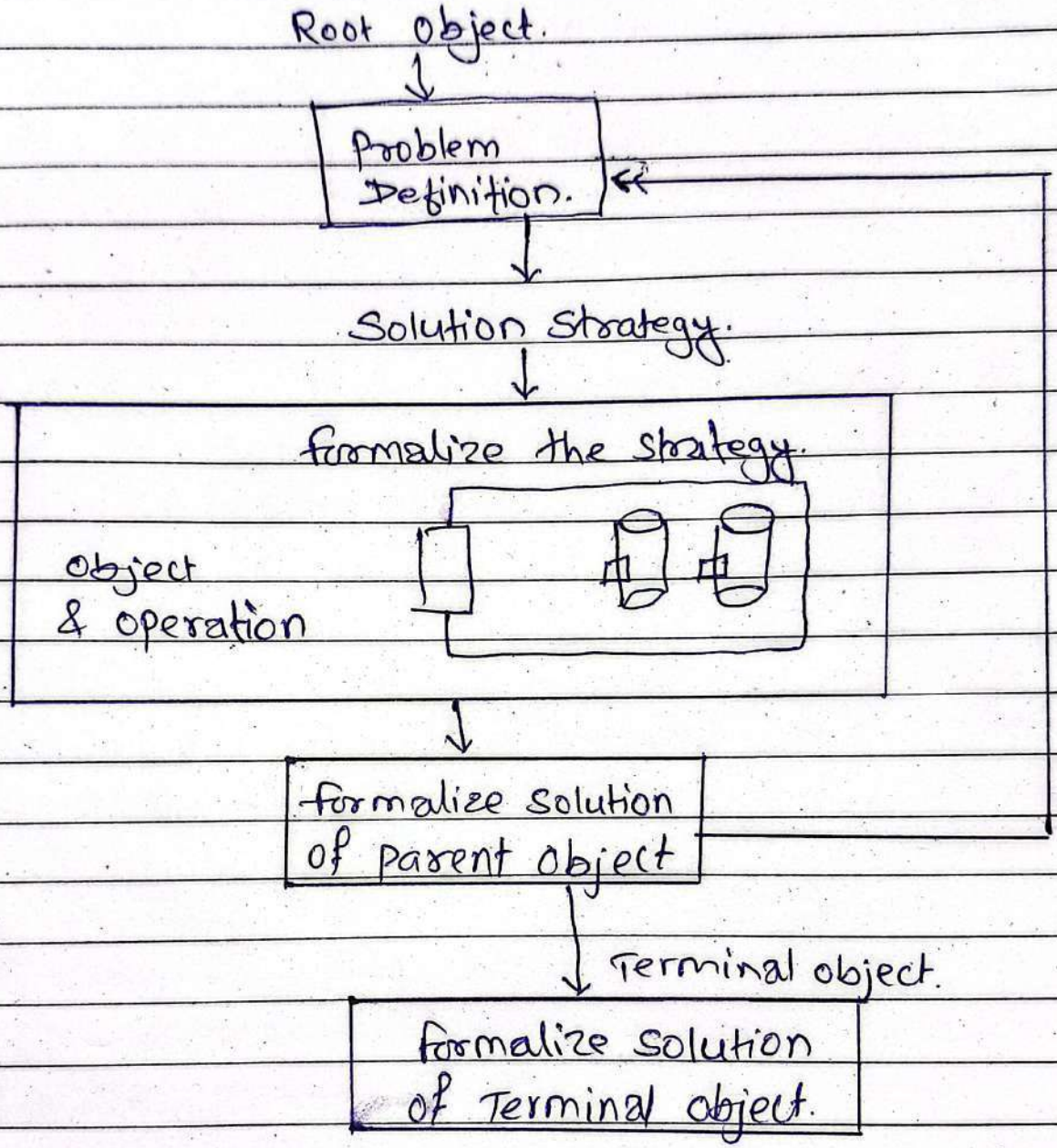


figure: Basic design steps in HOOD.

Requirement Engineering from User requirement Prespective:

Developing a requirement model from user requirement involves the process of systematically capturing, analyzing and documenting the needs of the users and stakeholders. It helps to create clear and comprehesive understanding of the system or product.

step 1: Requirement Elicitation:
→ gather info from stakeholder, their needs and expectation.
→ involves interview, survey, workshop and other forms of communication.

step : 2: Requirement Analysis:
→ Analyse requirement for any inconsistencies, ambiguity, priority or gap.
→ makes clear and accurate user's need.

Step : 3: Requirement specification:
→ Requirement is documented in structured way using templates such as use-cases, user stories, or functional requirement documentation.

Step 4: Requirement validation:
→ The documented requirement is validated with stakeholders including end-user, clients and the

development teams to ensure its accuracy, completeness and alignment with user needs.

Step: 5: Requirement Verification:
→ Validated requirements are now verified to check whether they are complete, consistent and un-ambiguous or not.

Step: 6: Requirement Communication:
The document is shared among developer teams, testers and other relevant stake-holde to ensure common understanding.

Step 7: Requirement management:
→ Track changes and updates to requirement throughout the project life cycle. Every changes must be documented.

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.

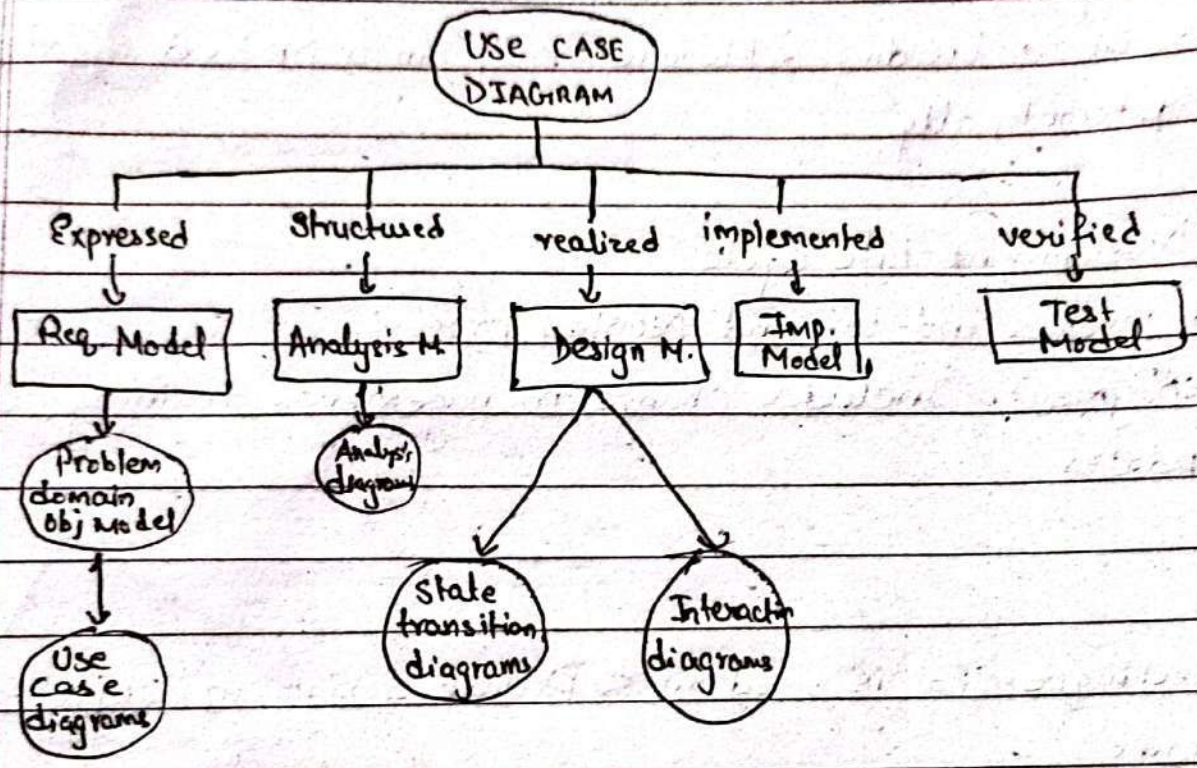


fig: Object-Orient SE

⇒ 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

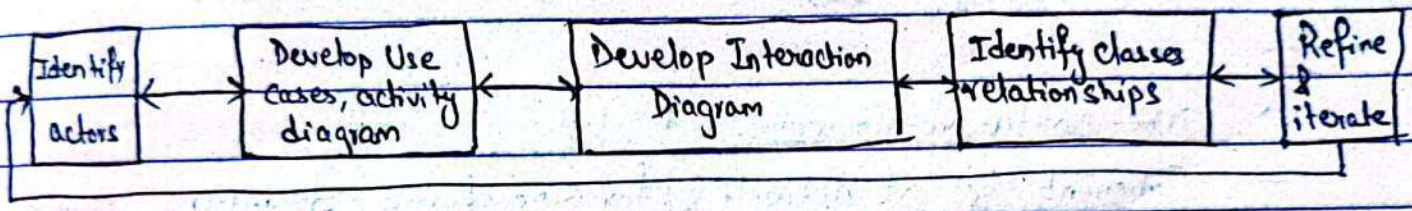


Fig: OO analysis

⇒ 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 relation-ships, 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

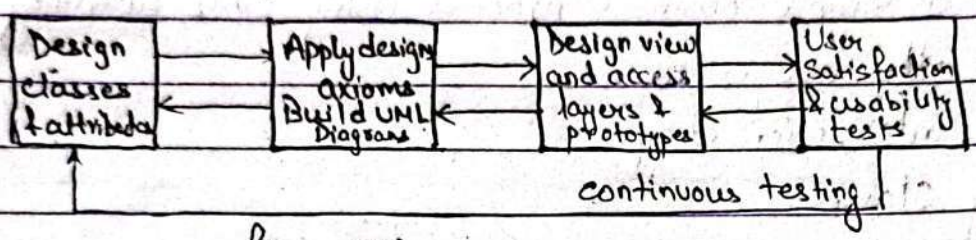


fig: OOD

⇒ 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

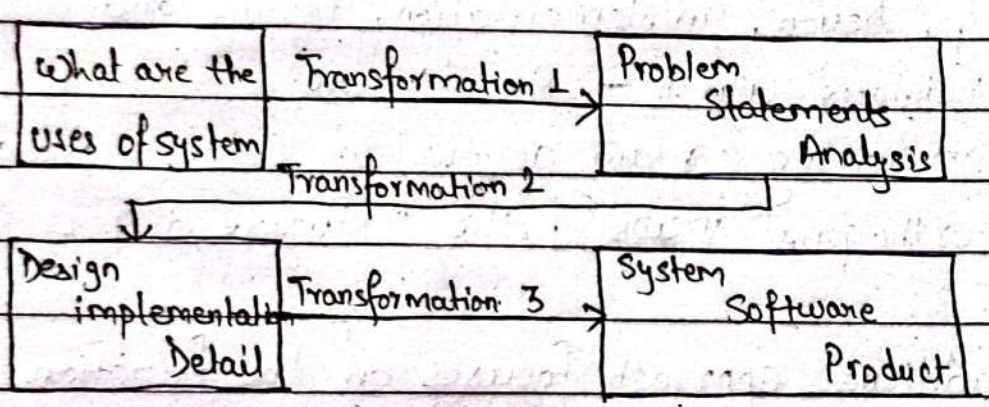
↳ Object Oriented system development centers on the object, which combines data and functionality.

-> 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.

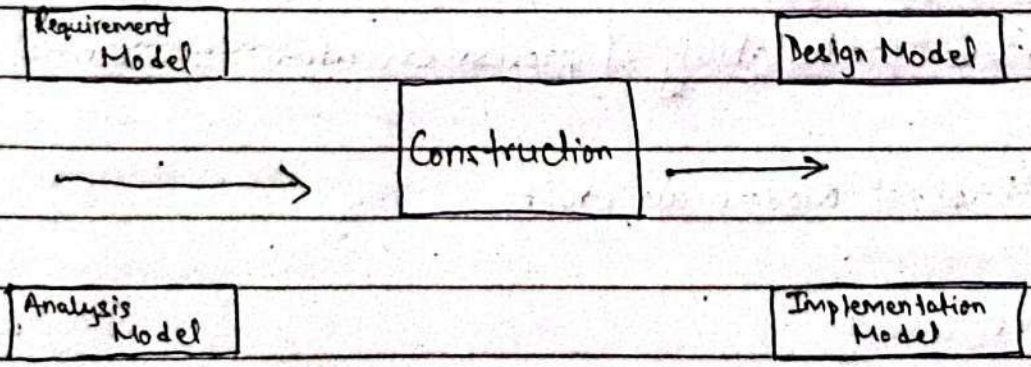
↳ Problem Domain object.
- → when requirements specification exists in very vague form, it is difficult to define task and boundary of a system.
- → It is good to develop a logical view of the system using problem domain object.

=> 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.

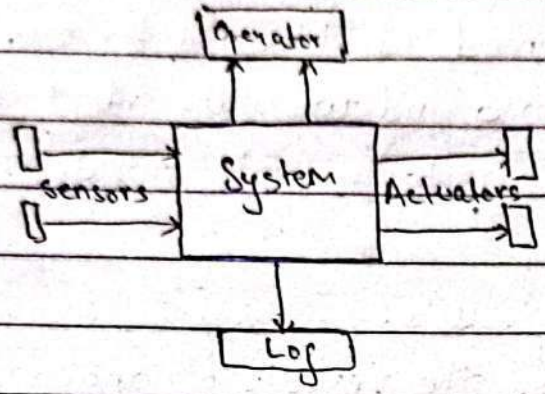


Fig: Abstract Model of RTS

→ 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.

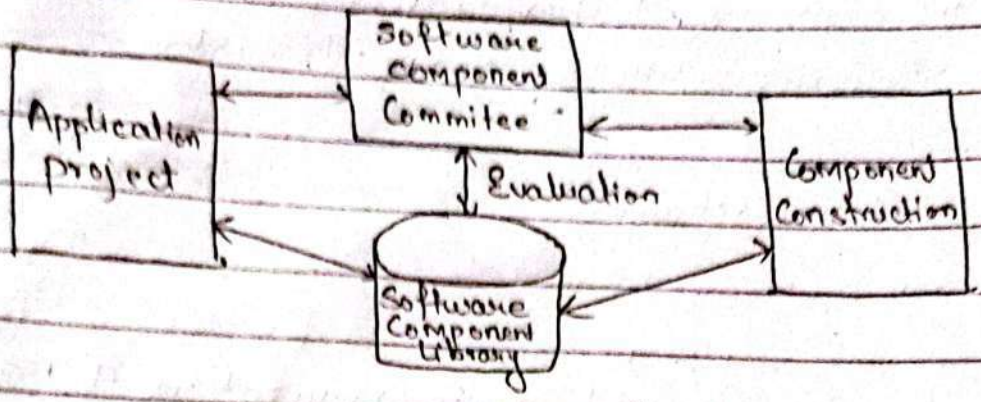


Fig: An organization of component management.

⇒ 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 ~~m~~ 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 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 → Evolution/Implementation → Maintenance

Established using context diagrams & prototypes

Describes roles and responsibilities of object using class diagram

Classes & relation about them using cla diagram

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, department 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 would undertake.

↳ The most viable options need to be chosen;

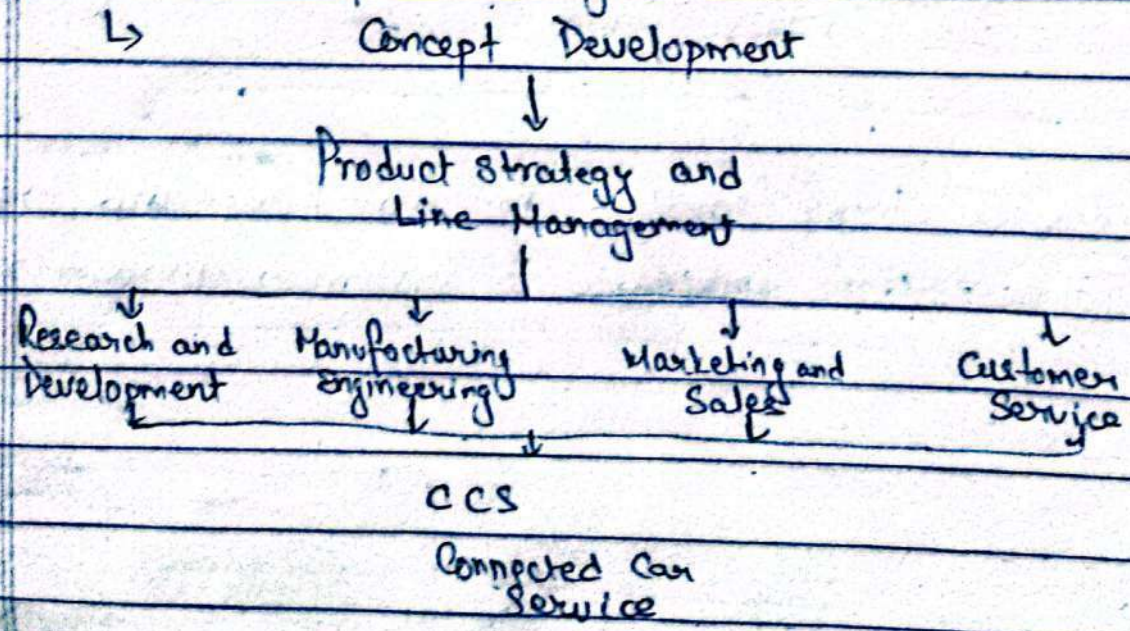
i) keeping in mind the goals and requirements of organization.

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.

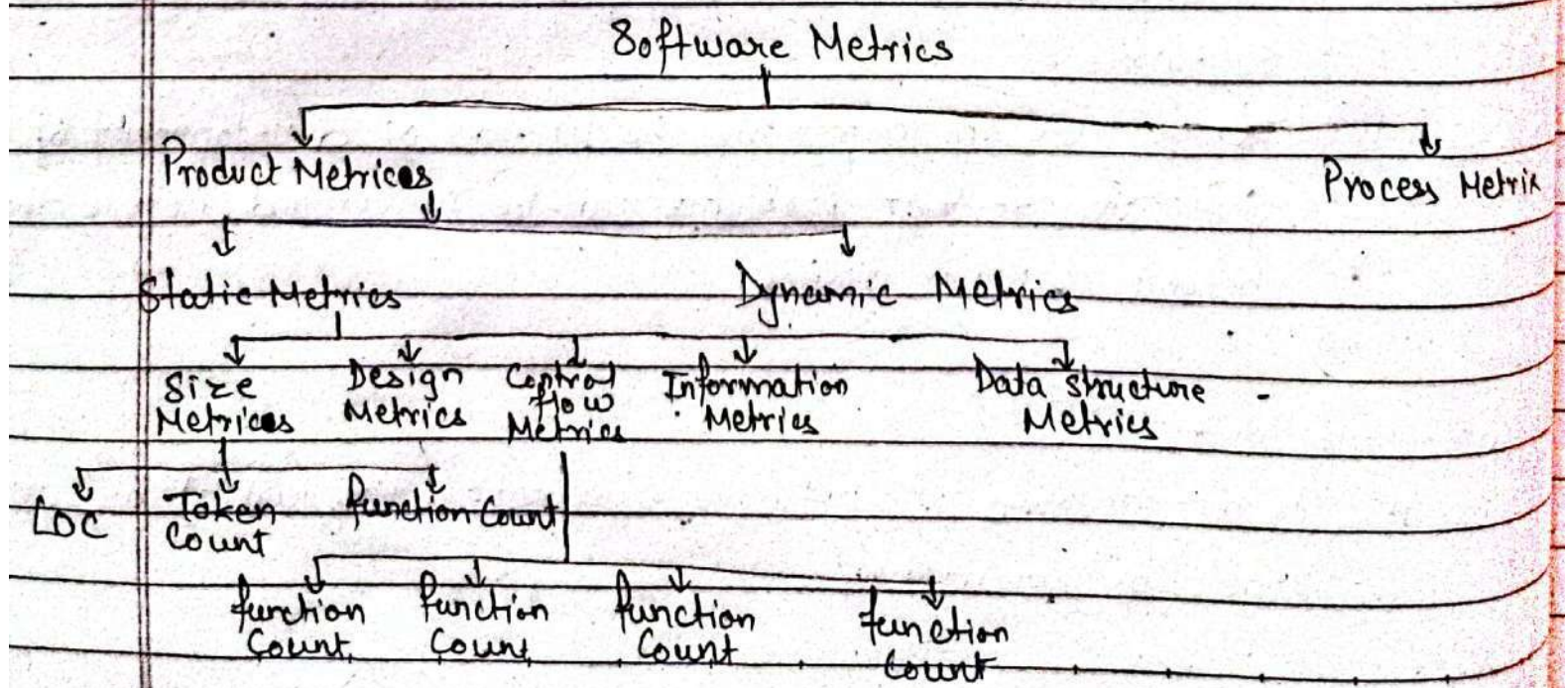
↳ Two important software characteries are:

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