Unit III Software Design.

Design process - Design
Concepts Design model - Design
Heuristic - Architectural desig
Architectural styles - Architectu
Mapping using Data flow -
User interface Design: Interf
Analysis - Interface design
Component Level Design -
Designing class based Compon
traditional components.

UNIT III

SOFTWARE DESIGN.

Design Process is a process used to transform user requirements into some suitable which helps the programmers in Software coding and implementation.

Software design Levels.

(1) Architectural design.

(2) High Level "

(3) Detailed "

(4) Three characteristics that serve as a guide for the evaluation of a design.

Design

(i) Abstraction

(ii) Software architecture

(iii) Verification

(iv) structured partitioning

(v) Object oriented De

(vi) Modularity.

(vii) Information hiding.

(viii) Control hierarchy.

(ix) Functional Indepen

(x) Design classes

(xi) Refinement

(xii) Concurrency.

(xiii) Data structure.

(iv) Refactoring

c) Abstraction.

1. It higher level of abstraction.
2. The lower Level of Abstraction.

Abstraction Mechanisms.

1. Functional Abstraction.
2. Data "
3. control "

Criteria for an effective modular System.

1. Modular Decomposability.
2. Modular Composability.
3. Modular user stability
4. Modular protection.

is a process of elaboration which causes the designer to elaborate the original statements.

Software Architecture.

properties of Architectural Design.

1. Structural properties.
2. Extra Functional "
3. Reusability.
4. Structural Models.
5. Framework Models.
6. Functional "
7. Dynamic "
8. Process "

5. Concurrency.

The software System can be categorized as Sequential or Concurrent Systems.

6. Information hiding.

is the fundamental design concept for software.

7. structure (or) Control hierarchy.

program Structure

Control hierarchy represents the organization of program components and implies a hierarchy of Control.

8. Verification:

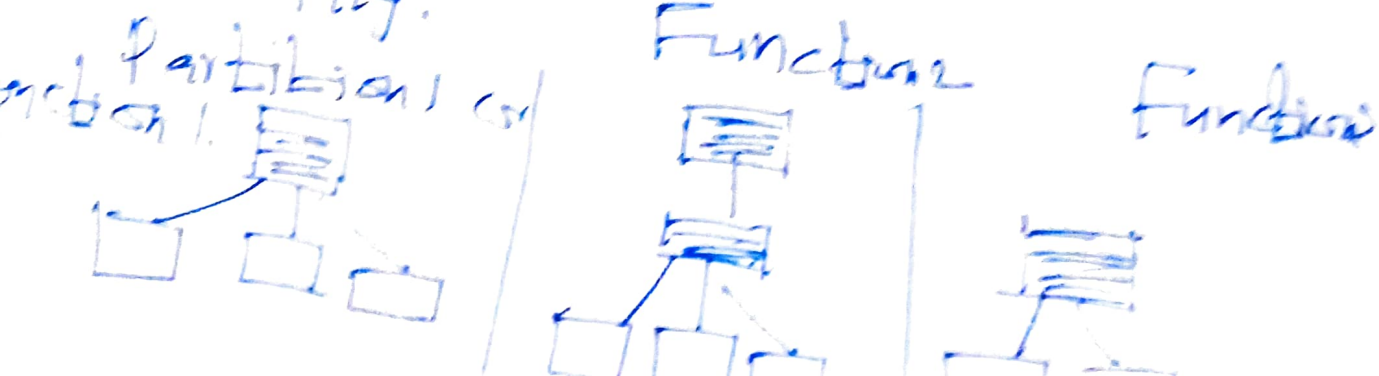
Verification is a fundamental concept of any design.

Verification is done in two steps.

1. Verification of the requirements
2. Verification of the design

9. Structured Partitioning

The program structure can be partitioned both horizontally and vertically.



is the direct outgrowth of Modularity and the concepts of abstraction and Information hiding.

11. Object Oriented Design concepts:

— is widely used in Modern software Engineering.

12. Design classes.

(1) User interface classes
2. Business domain "
3. process "
4. persistant "
5. System "

Component Level design.

– is a set of detailed drawings for each room in a house.

Deployment Level design e

– indicate how softw functionality and sub syste will be allocated within t physical computing environm that will support the softw

Design heuristic

1. Maintain Modularity
2. Fine tuning of the stru with respect to clarity an

3. Retain the scope of effect of the module within the scope of control of that module

4. Evaluate module interface to reduce complexity.

5. Define modules whose function is predictable

6. Strive for controlled entry modules by avoiding pathological connections.

Architectural Design.

1. Introduction
2. Software Architecture

Advantages.

Stakeholder Communicati

(4) System structuring
control Modeling.
(5)
(6) Modular decomposition
(7) The output of Archit
design process.
(8) Static structural Mod
(9) dynamic process
(10) Interface model
(11) Relationship :

Data Design

creates a model
data that is represented
User view of data which
higher level of abstraction
Applications :

Characteristics of Data Warehouse

1. Subject Orientation.
2. Integration
3. Time Variance
4. Non-Volatility.

Data design at the component level.

- focuses on the representation of data structures that are directly accessed by one or more software components.

Data dictionary

Should be used to define both data and program design.

Architectural Genre.

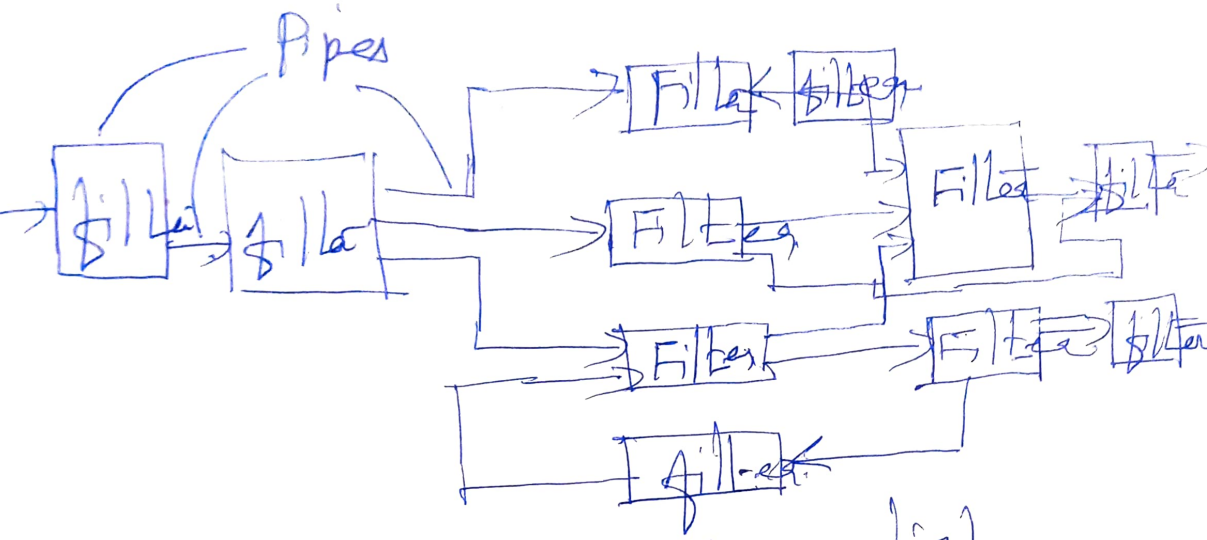
- Architectural intelligence
- Commercial and nonprofit communications.

1. Entertainment and sports
2. Games.
3. Platforms
4. Tools
5. Scientific
6. Transportation
7. Medical
8. Military
9. Utilities
10. Government.

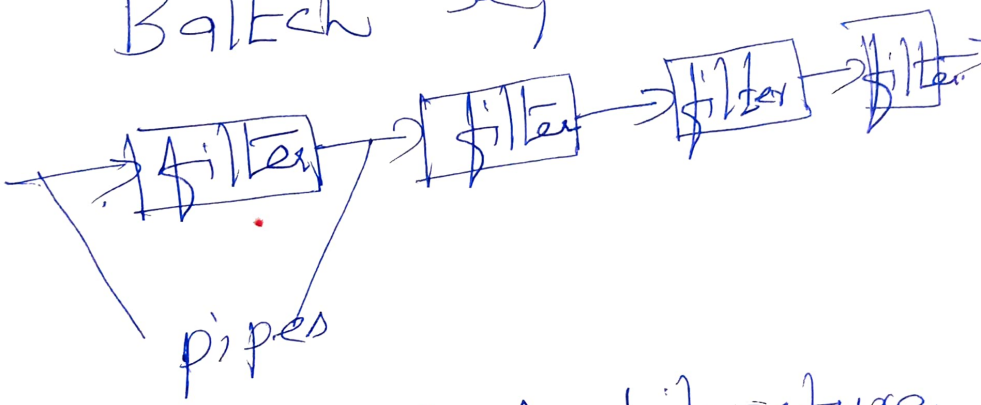
ARCHITECTURES STYLES.

Data centered Archit...

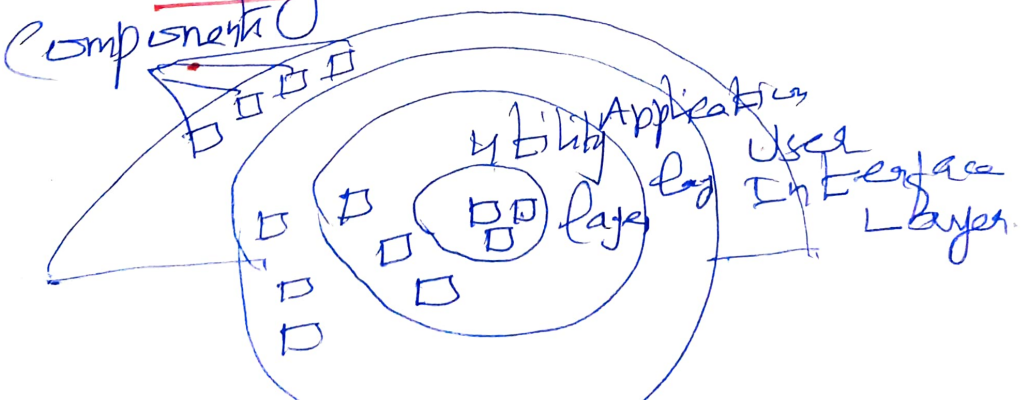
Pipes and filters



Batch Sequential



Layered Architecture



1. Representation in context

2. Defining Archetypes

Node

Detectors

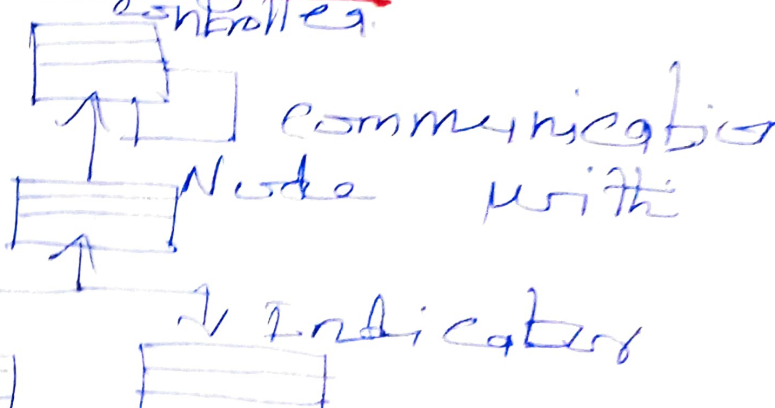
Indicators

Controllers

3. Refining the Architecture into components.

4. Describing Instantiation of the system

1) Defining Archetypes.



Architectural Mapping using DATAFLOW.

1. Transform Mapping.

Step 1. Review the fundamental System Model.

Step 2: Review and refine data flow diagrams for the software.

Step 3: Determining whether the DFD has transform transaction flow characteristi

Step 4: Isolate the transform center by specifying incomin and outgoing flow boundar

Step 5: Perform "First level factoring"

Step 6: Perform second factoring.

Step 1:- Review the fund... system model.

Step 2: Review and ref... flow diagrams for the ...

Step 3: Determine whether ... DFD has transform or tran... flow characteristics.

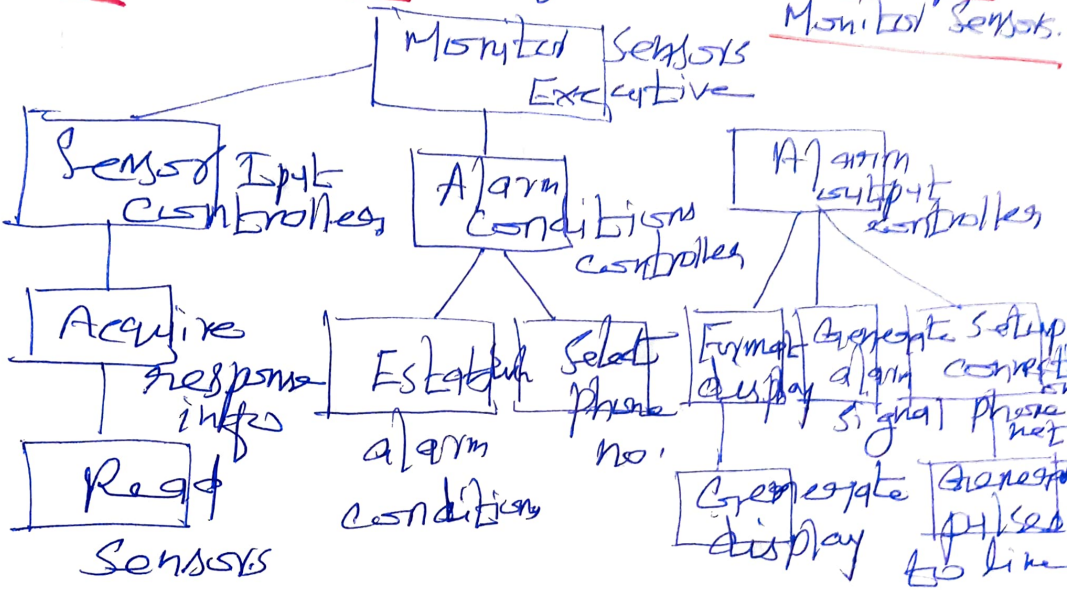
Step 4: Isolate the transform ... center by specifing incoming ... and outgoing flow bounda...

Step 5: Perform first le... factoring.

Step 6: Perform second ... level factoring.

Step 7: Refine the first iteration.

First iteration program structure for Monitor Sensors.



3.8. Transaction Mapping.

1. Design steps.

Step 1. Review the fundamental System model.

Step 2. Review and refine data flow diagrams for

the DFD has transform flow characteristics.

Step 4: Identify the transaction center and the flow characteristics along each of the action paths.

Step 5: Map the DFD in a program structure in transaction processing.

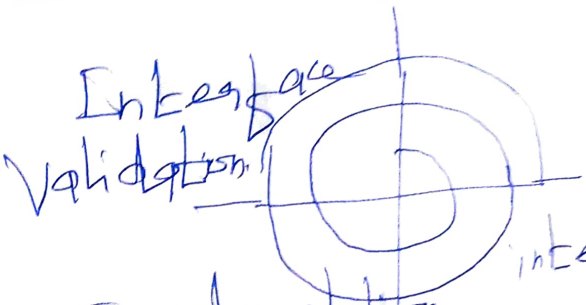
Step 6: Factor and refine the transaction structure and the structure of each action path

Step 7: Refine the first iteration architecture using design boundaries for improved software quality.

39. User Interface Design

1. Introduction

2. User interface Analysis and design



1. User interface design principles.
2. Place the user in control
3. Reduce the user's Memory load
4. Make the interface consistent

User Interface Analysis Steps

- User interviews
- Sales input
- support :
- task analysis and Modeling.

Task elaboration
Object elaboration
Workflow analysis
Hierarchical representation.

User task: Requests that a prescription is refilled.

- provide identifying information
- specify name.
- specify user id
- specify PIN and password.
- specify prescription number.
- specify date refill is required.

Analysis of Display content

The format and aesthetics of the content are considered.

Steps

1. Applying interface Design steps.
2. User Interface Design Pattern
3. Design Issues.
4. Response time
5. Help facilities
6. Error Handling.
7. Menu and command labeling
8. Application accessibility.
9. Internationalization.

Component level Design.

1. Overview.
2. Software component.

Block is a modular building for computer software different views

Object - Oriented view.

— A component is viewed as a set of one more collaborating classes.

- Conventional or Traditional

Conventional software components are derived from the (DFDS) in the analysis mo

- process - related view.
- class based component des

(I) class based component design principles.

(II) component - level desig Guidelines.

(III) cohesion

— is the single-min component

Coupling

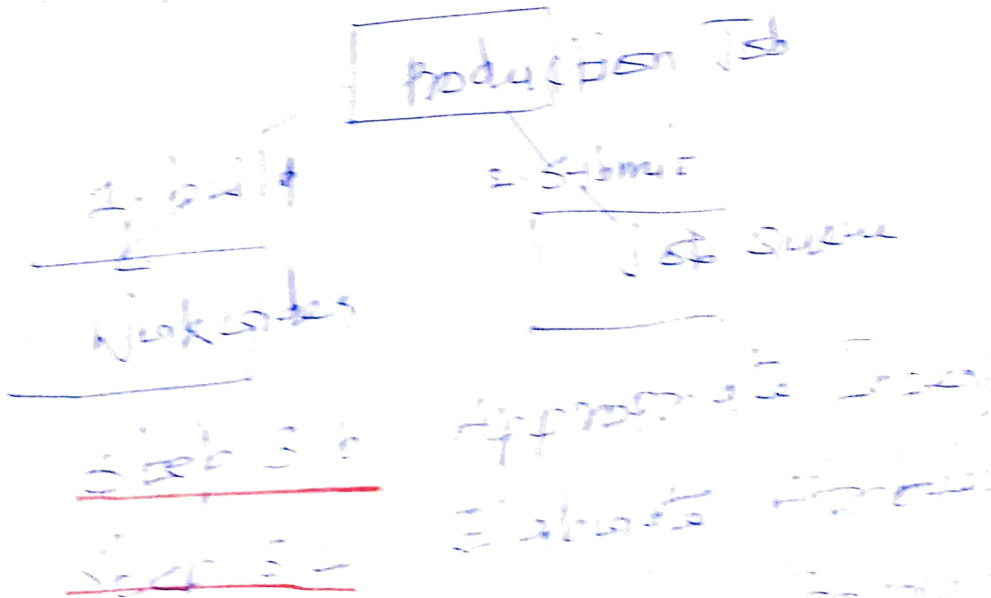
is a qualitative measure of the degree to which operati and classes are connected to one another.

- Conducting Component level design

Step 1 & Step 2 Identify classes

Step 3 - class Elaboration

Step 3a - Collaboration details



Step 4 - Persistent data.

Step 5 - Elaborate Behavio

Step 6 - Elaborate Deploym

Step 7 - Redesign / Reconsider

* Component level design WebAp

1. - WebApp Component level de

2. " " " content

design

3. WebApp component level

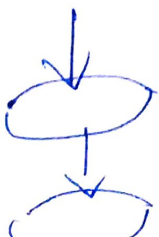
Functional Design

4. Traditional or Conventional

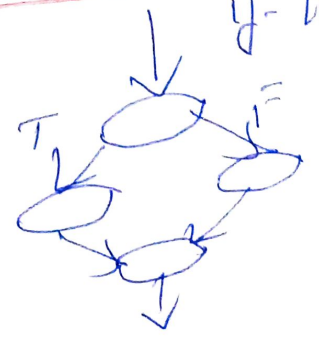
Component level design.

Design Notation.

1. sequence.



if-then



Domain Analysis steps

1. Define
2. Categorize
3. collect
4. Analyse
5. Develop

Component Qualification, Component Adaptation and Component Qualification.

Component Adaptation.

- White box wrapping
- Grey box "
- black box "

Component Composition

Architectural Ingredients

- Data exchange model.
- Automation.
- Structured storage.
- Underlying Object Model.

Analysis and Design for Reuse

The requirements mod
needs to be analysed to dete
the elements that point to exist
reusable components.

Classifying and Retrieving Components.

Describing reusable compon

Role environment elements.

- a Component database - Storing software components
- Library Management - to allow access to the database.
- Software Component retrieval system
- CASE tools that support the integration of reused components into a new design or implementation

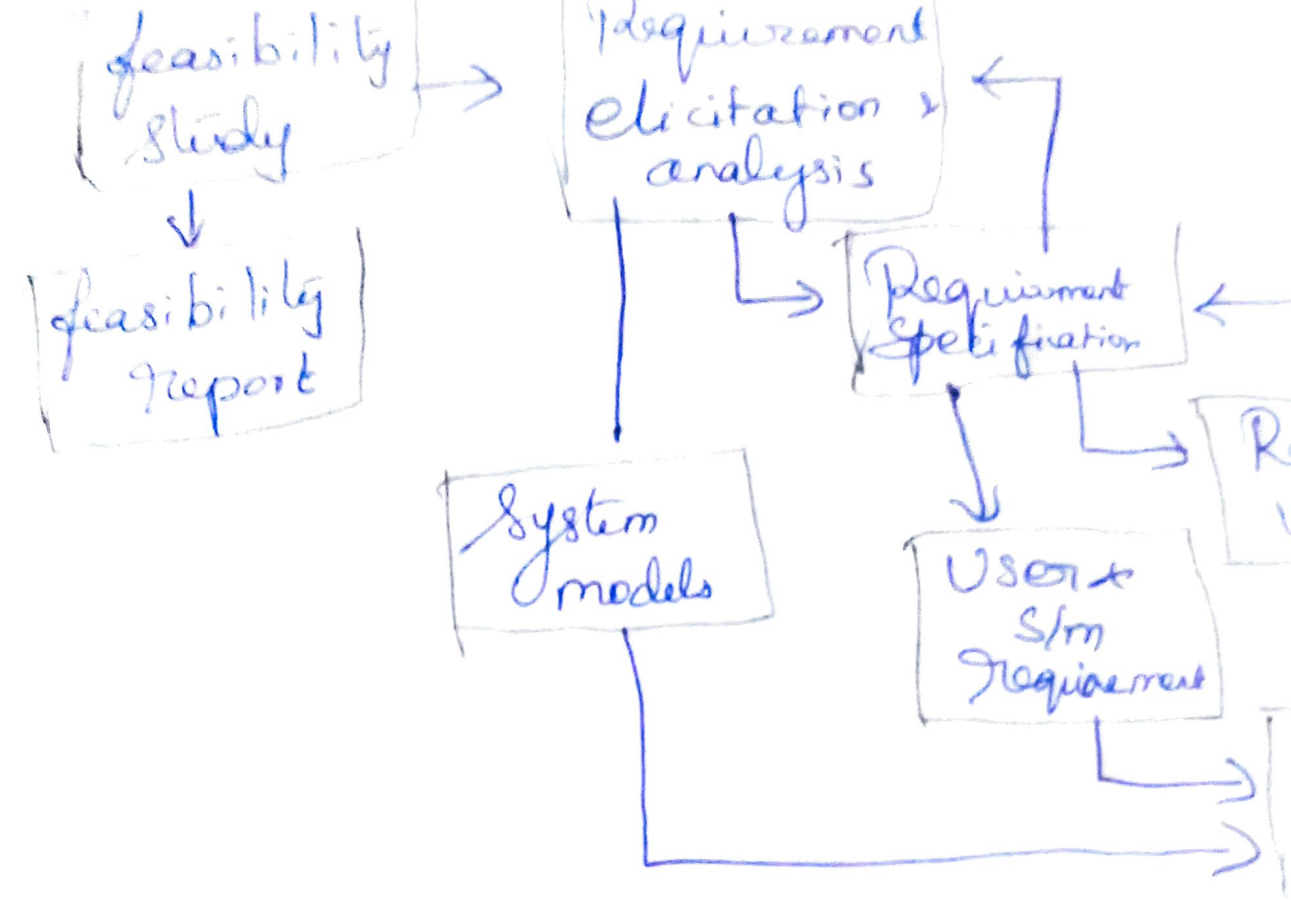


fig: Requirement engineering Process