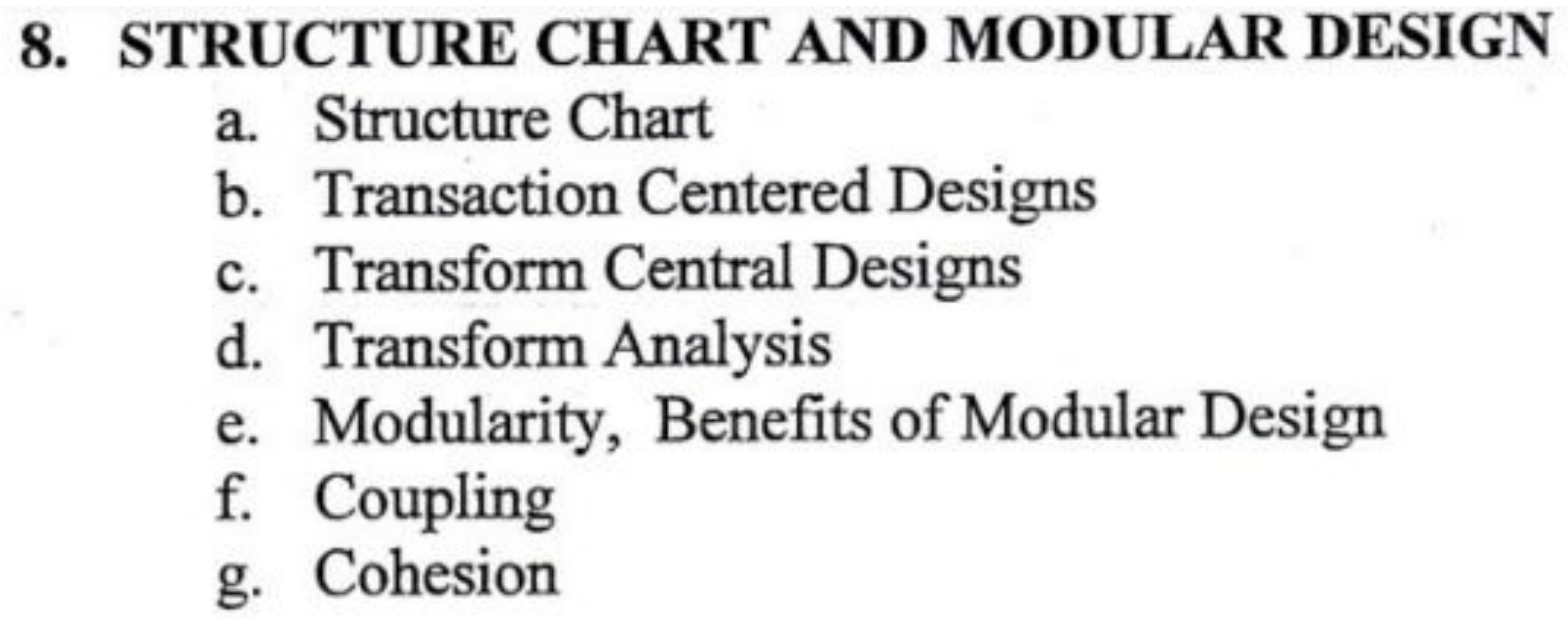
Chapter 8

Structure Chart and Modular design

Contents



8. STRUCTURE CHART AND MODULAR DESIGN

a. Structure Chart
b. Transaction Centered Designs
c. Transform Central Designs
d. Transform Analysis
e. Modularity, Benefits of Modular Design
f. Coupling
g. Cohesion

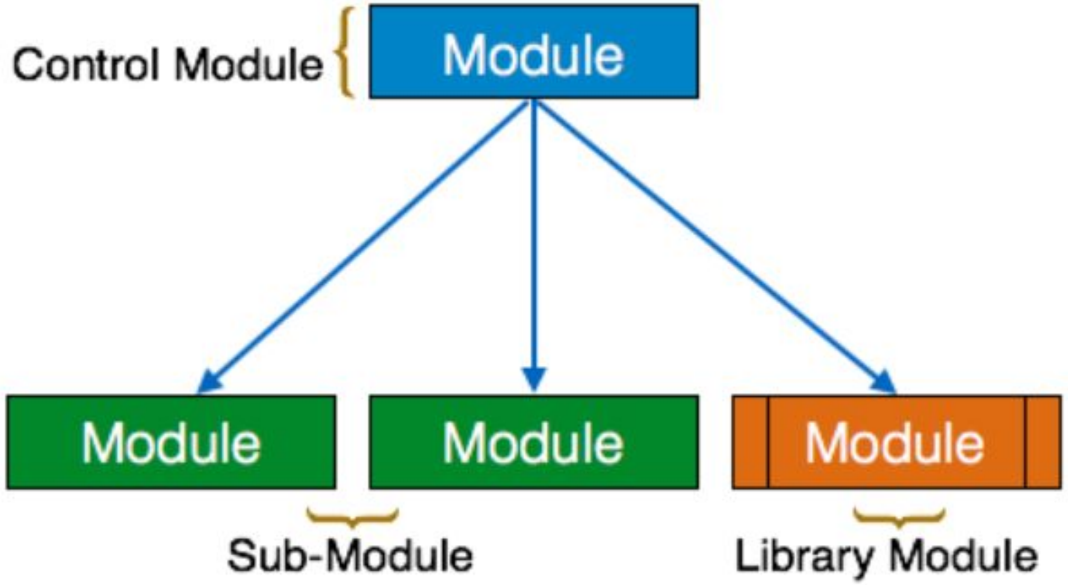
8.1 Structure chart

- Also known as **module chart** or **hierarchy chart**.
- Shows how an information system is organized in a hierarchy of components ,called modules.

Symbols used in the structure chart:

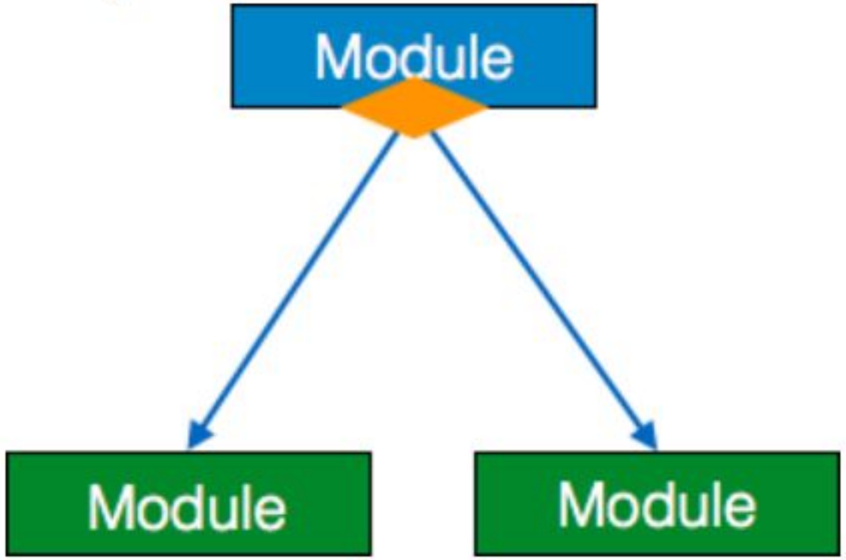
1. Module:

- A small unit of a system ,defined by its function.
- Different types of modules are:
  - □ A **control module** , a higher-level module , branches to more than one **subordinate-module**.
  - □ **Library Modules** are re-usable and invokable from any module.



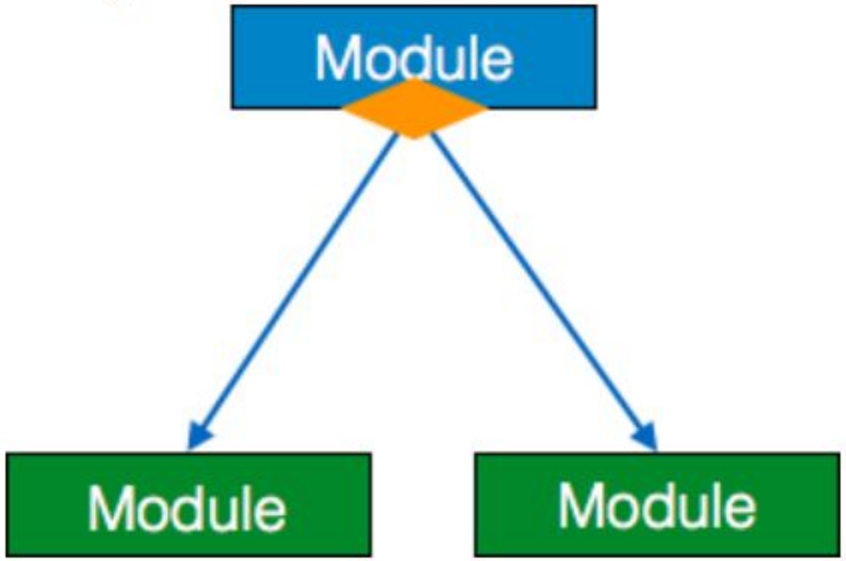
2.Condition

- It is represented by small diamond at the base of module.
- It depicts that the control module can select any of sub-routine based on some condition.



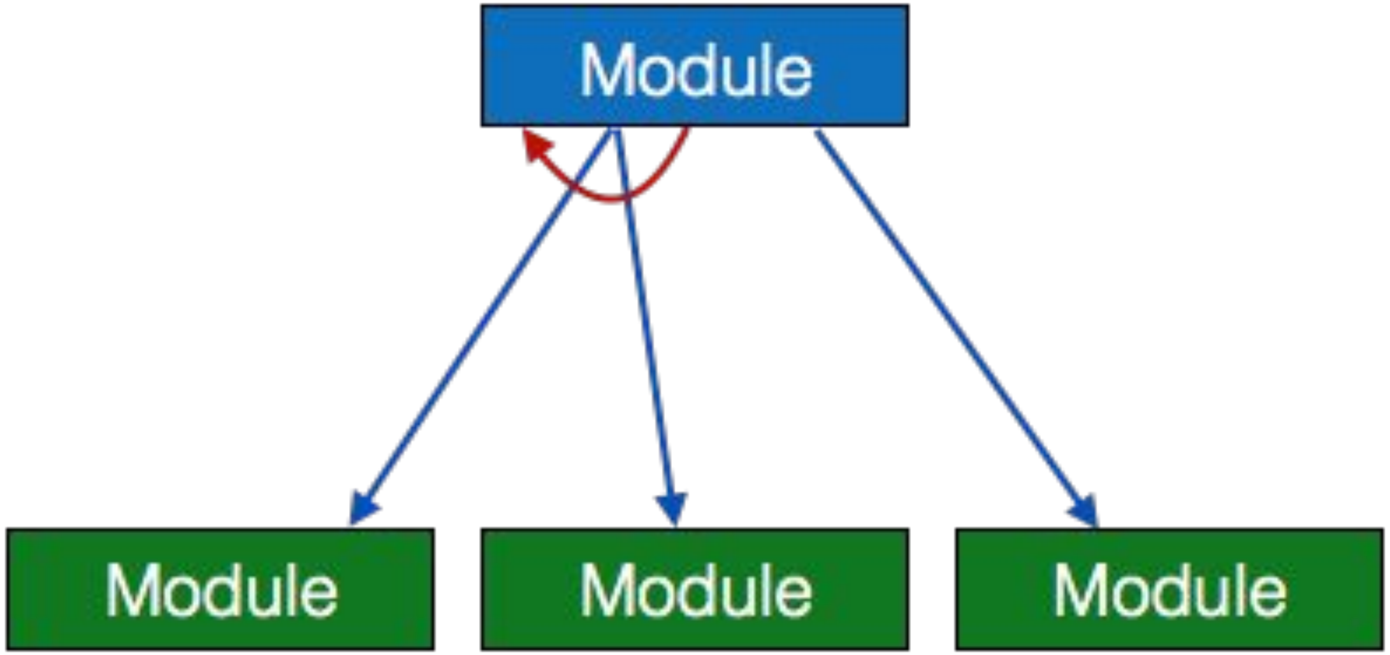
3.Jump

- An arrow is shown pointing inside the module to depict that the control will jump in the middle of the sub-module.



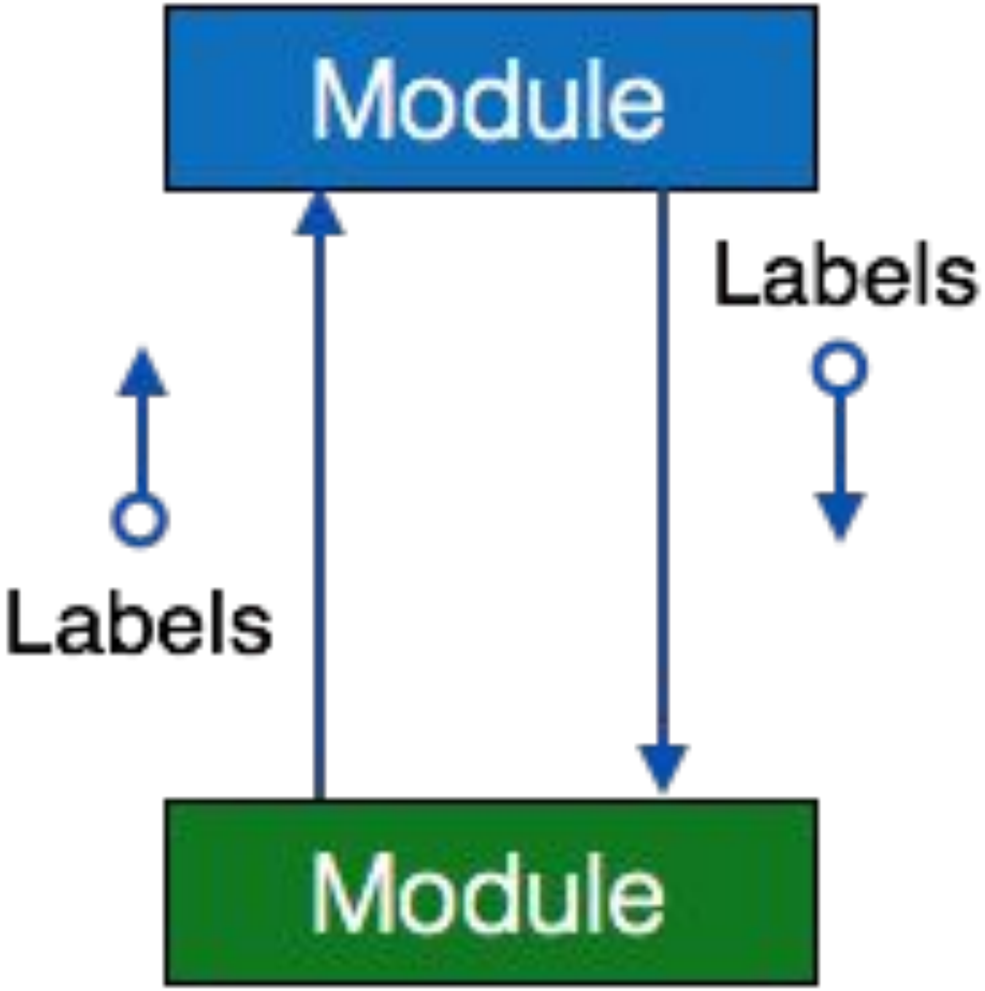
4.Loop

- A curved arrow represents loop in the module.
- All sub-modules covered by loop repeat execution of module.



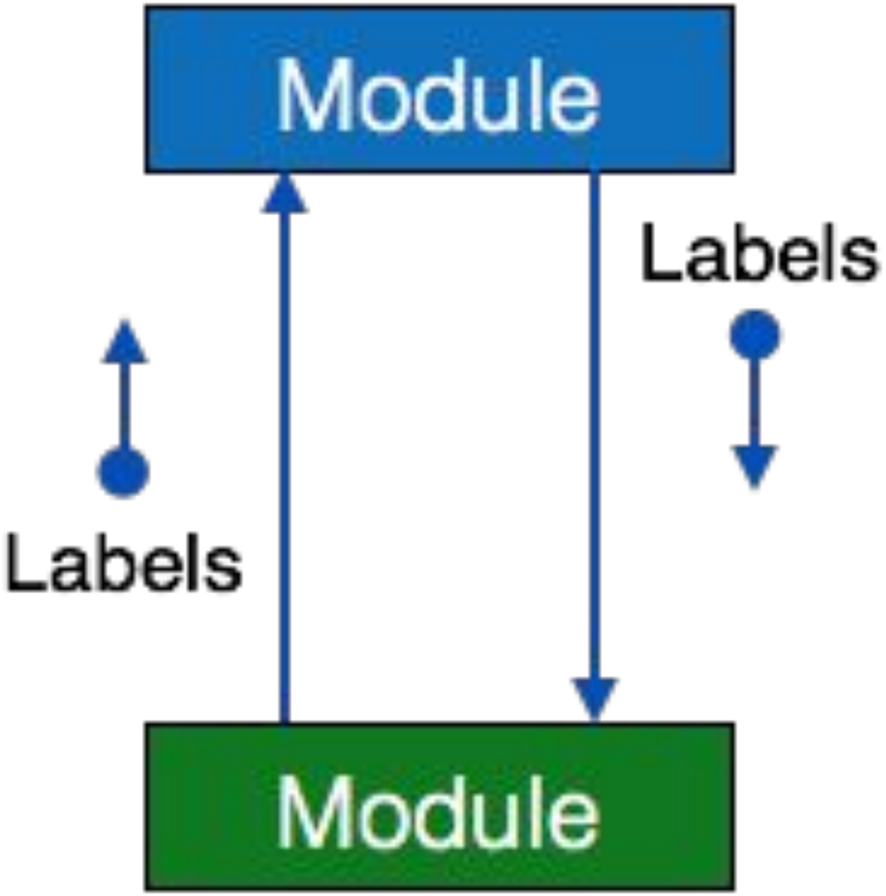
5.Data flow

- A directed arrow with empty circle at the end represents data flow.

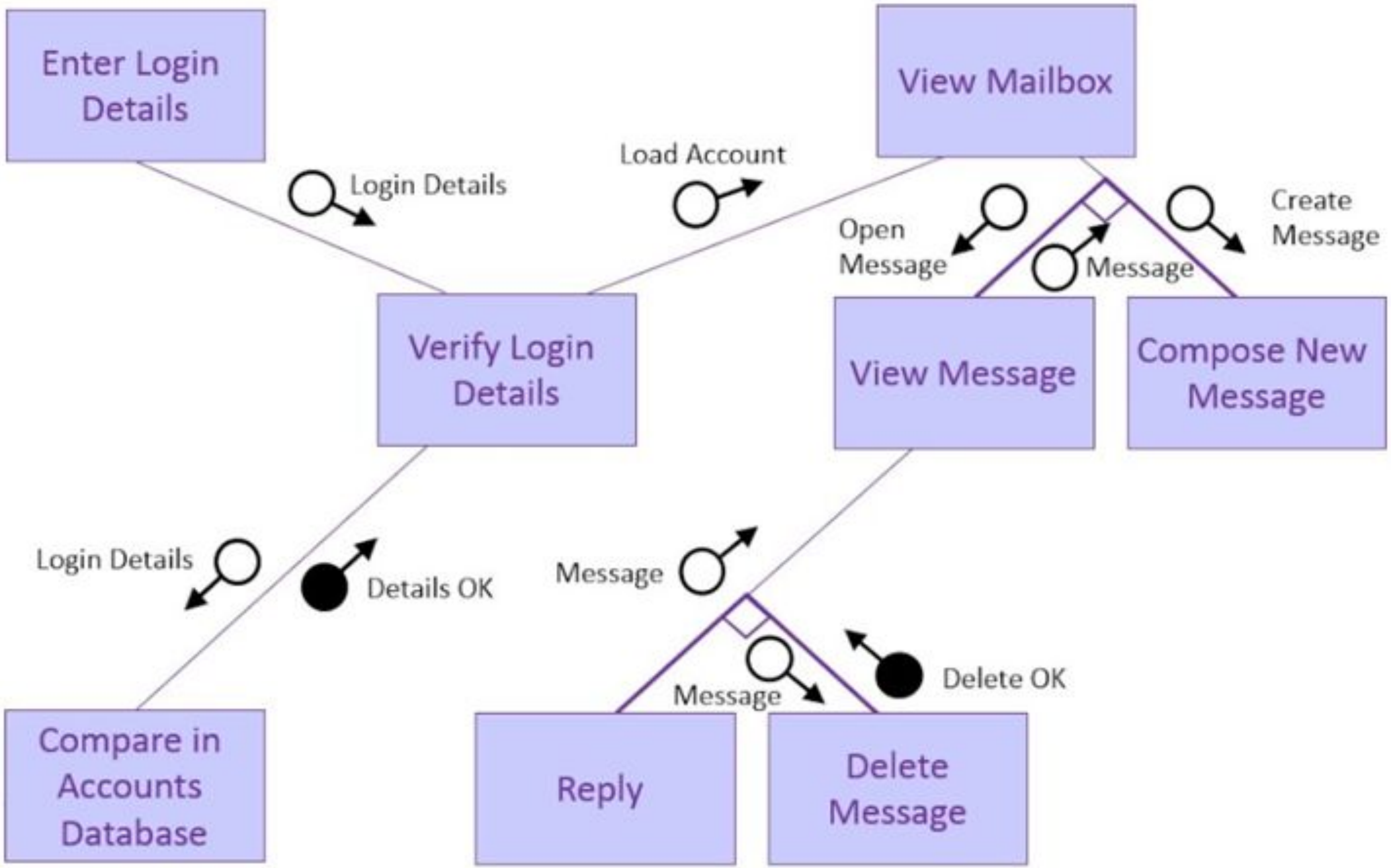


6.Control flow

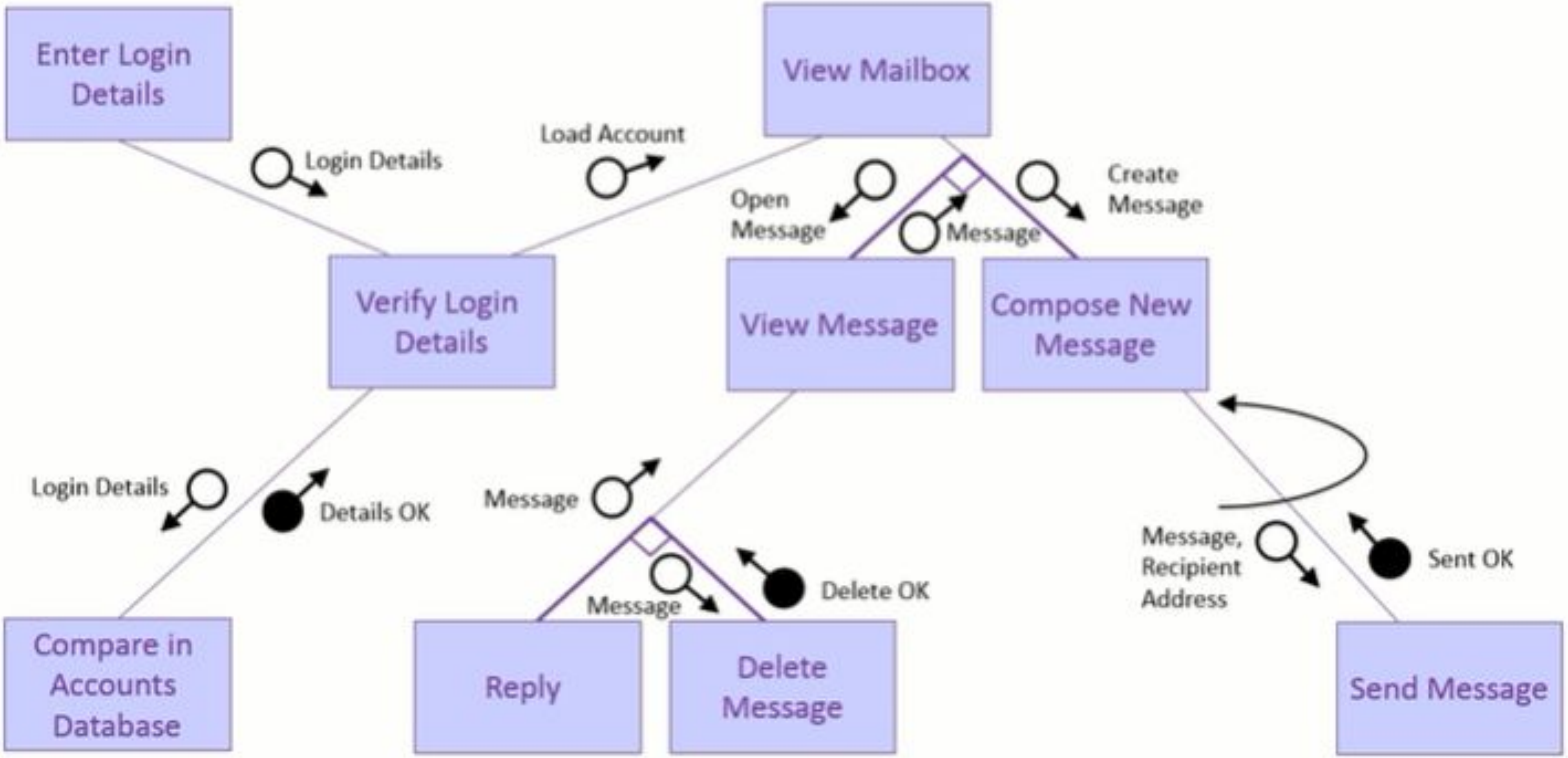
- A directed arrow with filled circle at the end represents control flow.



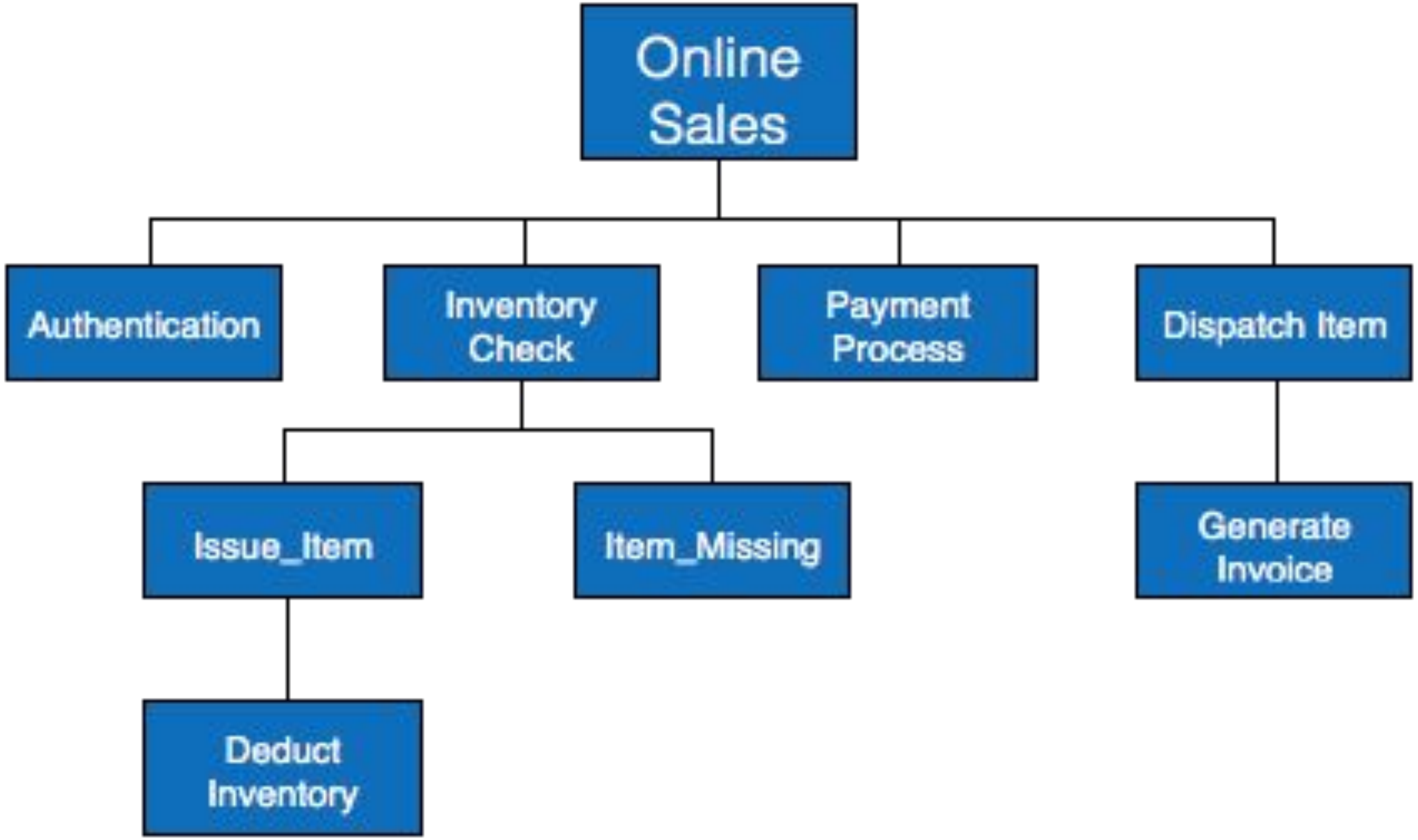
Example1: A structure chart of message system.



Example: The following Structure Chart outlines the use of an Email Server



Example3: Structure chart of online sales system.

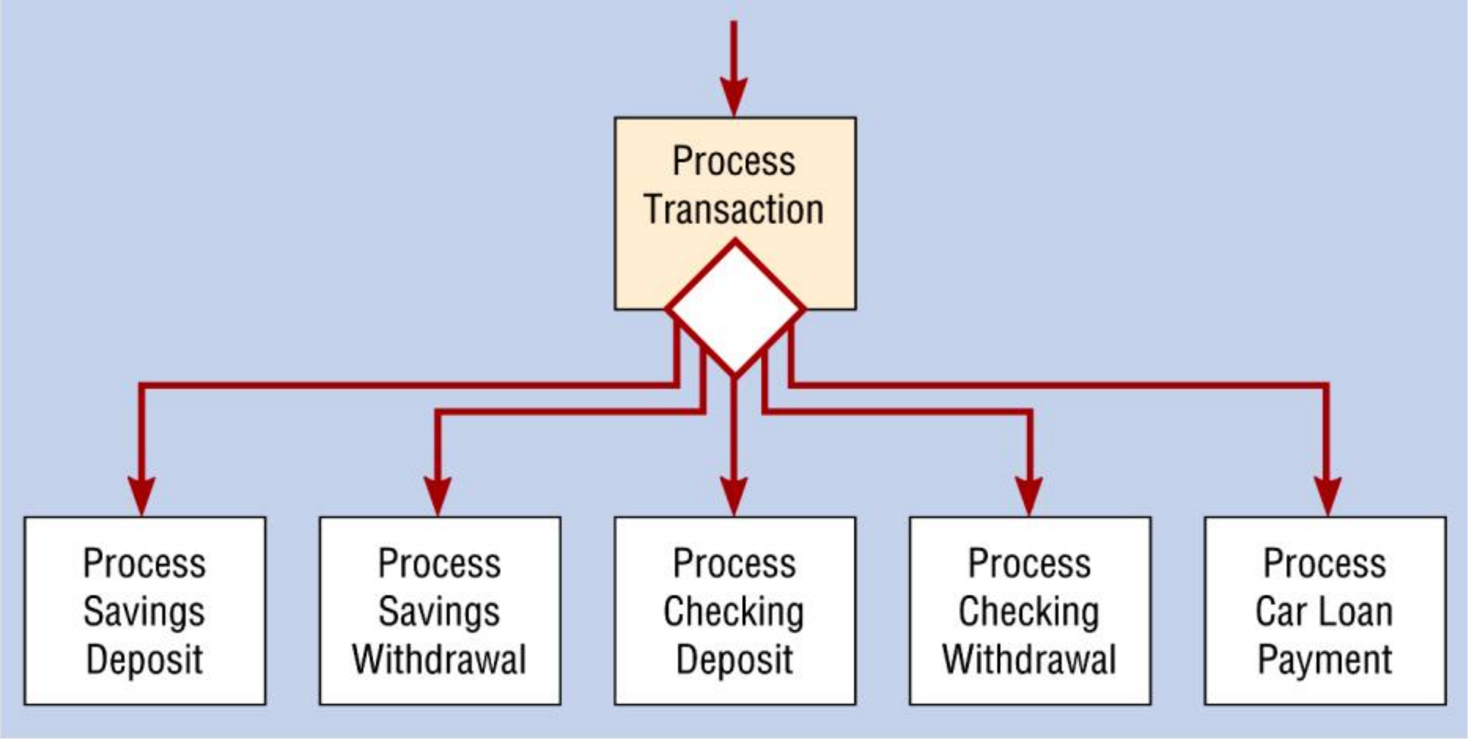


8.2 Transaction Analysis Design

- Structure chart can be designed under following headings:
    1. Transaction Centered Designs
    2. Transform Centered Designs

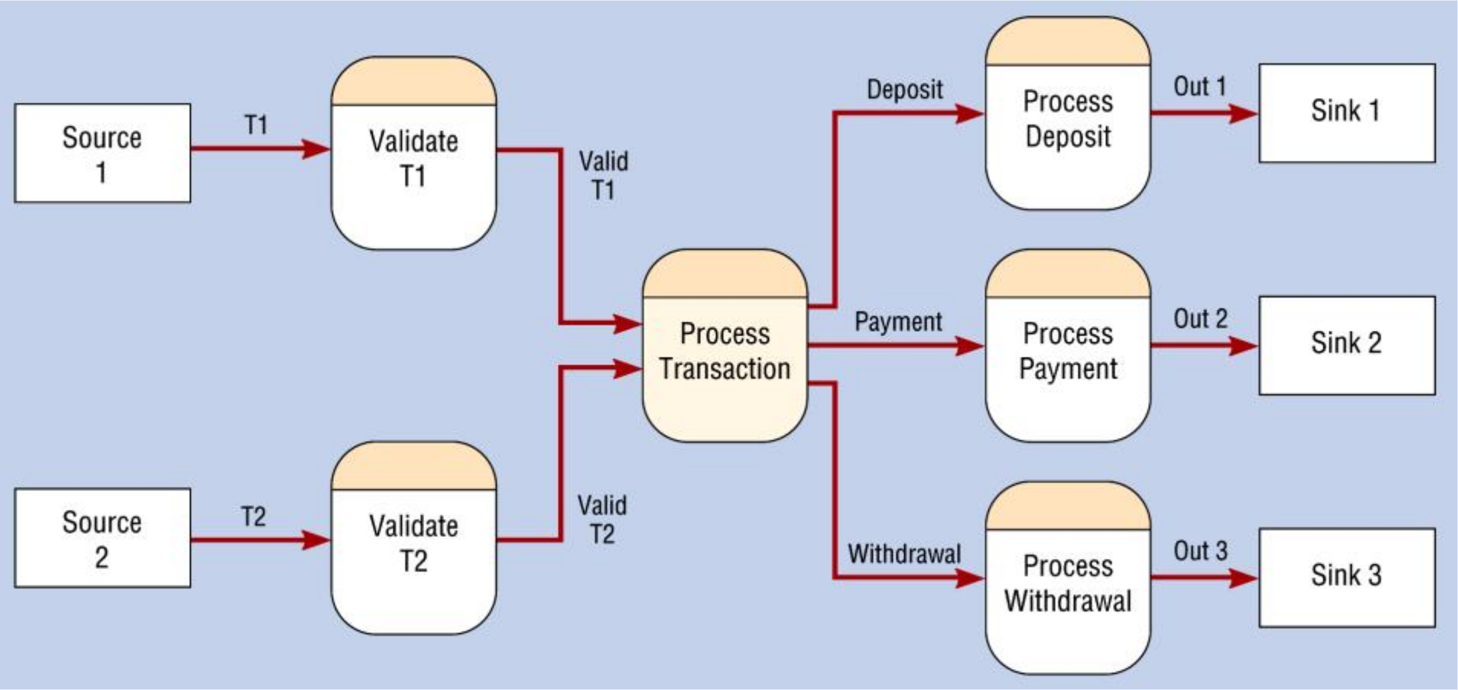
1. Transaction Centered Designs

- Data come in central module in the system and dispatch to their proper location based on data type.



2.Transform Central Designs

- Focus on the derivation of new information from existing data.



8.3 Transform Analysis

- The process of converting the DFDs of transform centered system into their corresponding structure chart.
- Consist of TWO main parts:
    1. TOP level design
    2. Detailed design

1.Top level Design:

I. **Identify afferent and efferent flows**
   - The input part of the system is known as afferent flows and
   - The output part of the system is known efferent flows.

II. **Find central module**
   - Trace each afferent flow forward until it disappears and
   - Trace each efferent flow backward until it disappears.
   - The point at which all the flows disappears is a central module.

III. **Identify Boss or coordinating module**
   - Monitors each modules in the system.

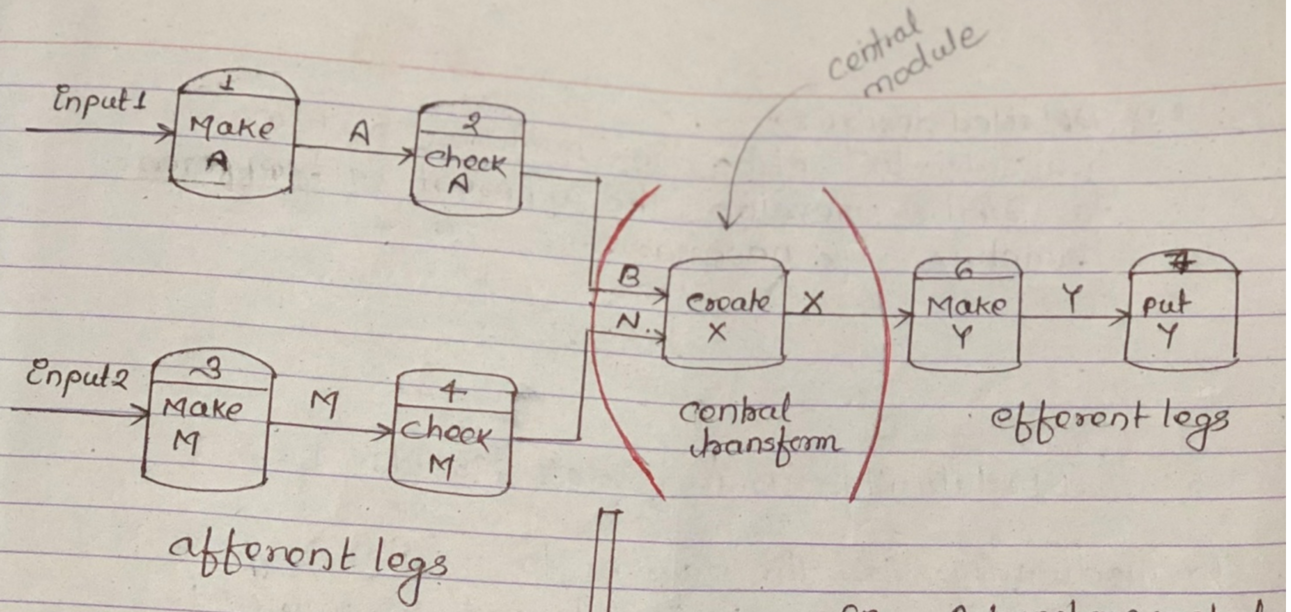


fig : A transform central design.

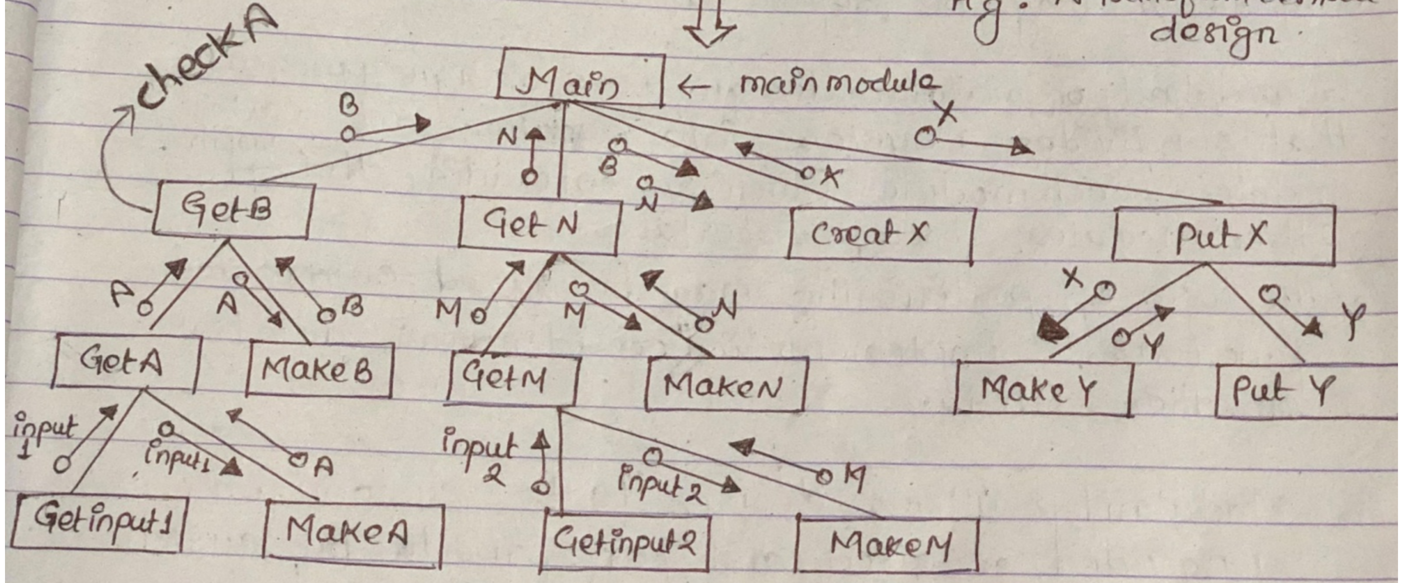


fig: An equivalent structure chart of above

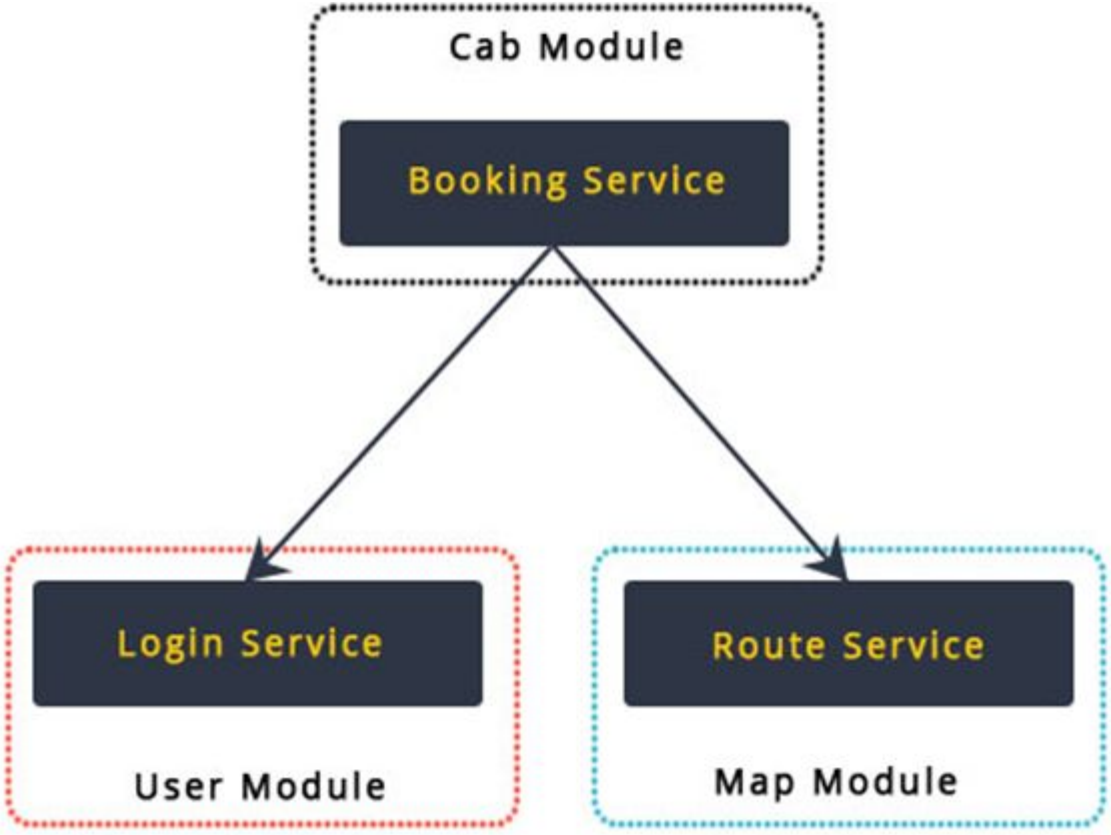
2.Detailed Design

- In order to refine the structure chart , we need to further develop the afferent and efferent branches if necessary.

8.4 Modularity

- The degree to which a system’s components may be separated and recombined is called modularity.
- A design approach where a system is broken down into subsystem or modules.
- Each module functions interacting with other modules through the interface.
- Modularity is a strategy for organizing complex products and processes efficiently.

Example:


Cab Module
Booking Service
Login Service
User Module
Route Service
Map Module

Advantages

- Independence, Reuse, and Efficiency in Modular Design.
- Modular Doesn’t Mean Boring.
- Modularity Leads to a Consistent Design .
- Improves system maintainability.

Disadvantages:

- Does not work well for asynchronous system.
- Could become complex for large system.

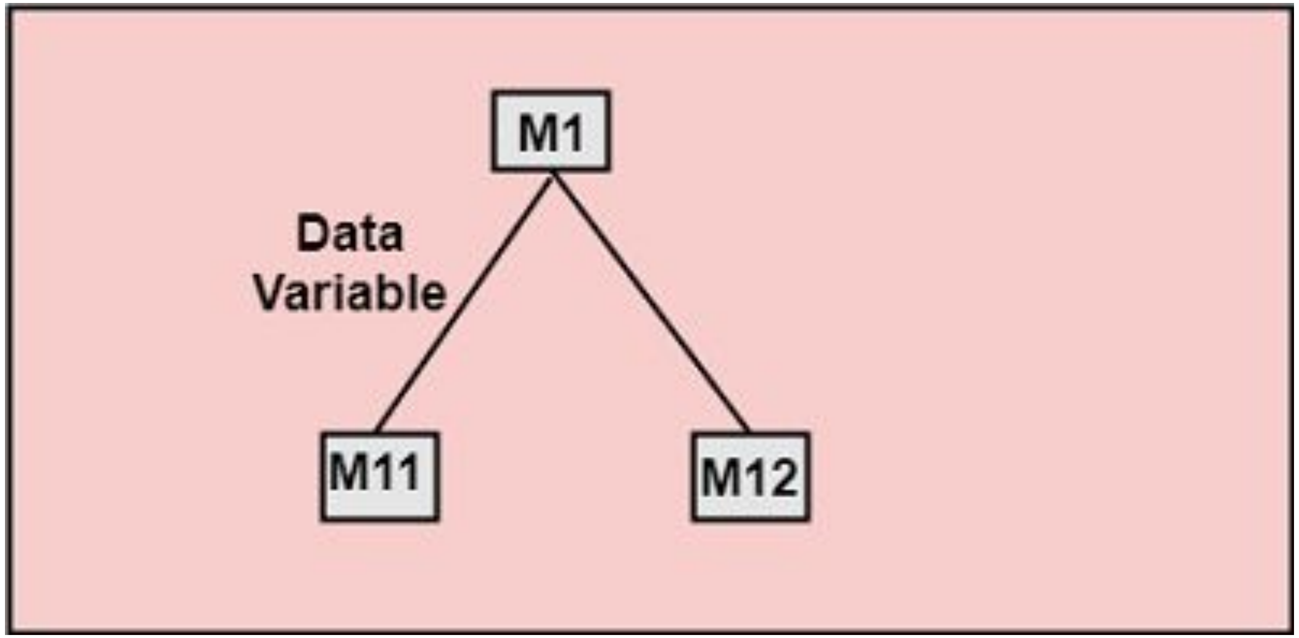
8.5 Coupling

- Coupling is the measure of the degree of interdependence between the modules.
- A good software system will have low coupling.
- **SIX** Types of coupling

  1. Data coupling
  2. Stamp coupling
  3. Control coupling
  4. Common coupling
  5. Content Coupling

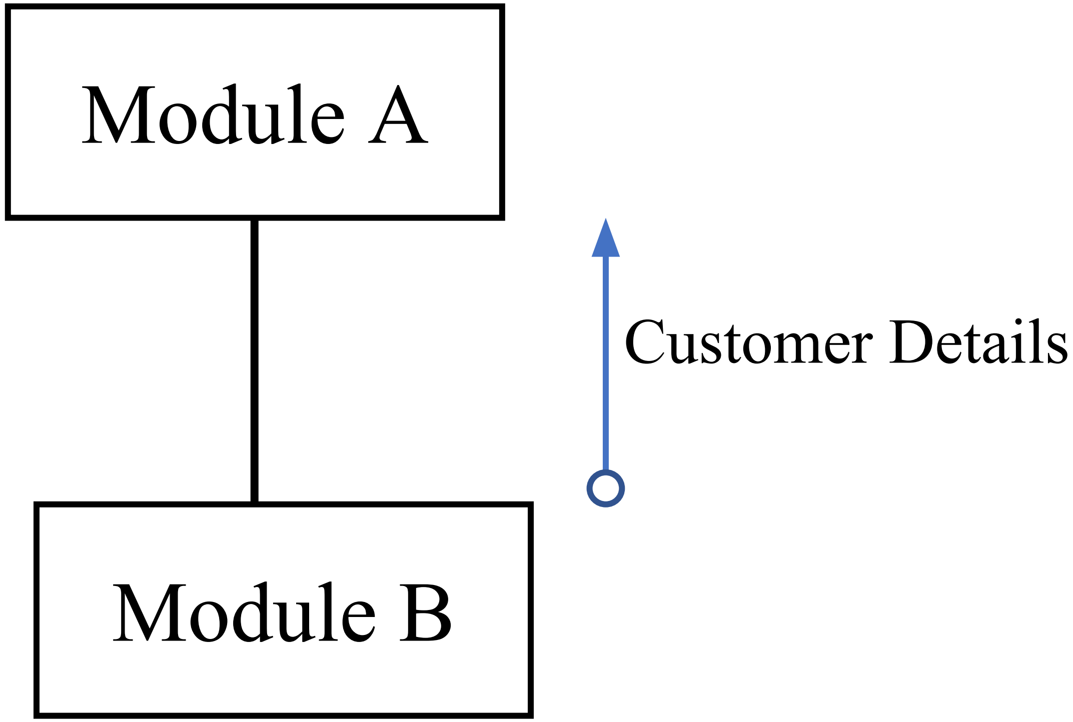
1.Data Coupling

- When data of one module is passed or shared to another module, then this is called data coupling.



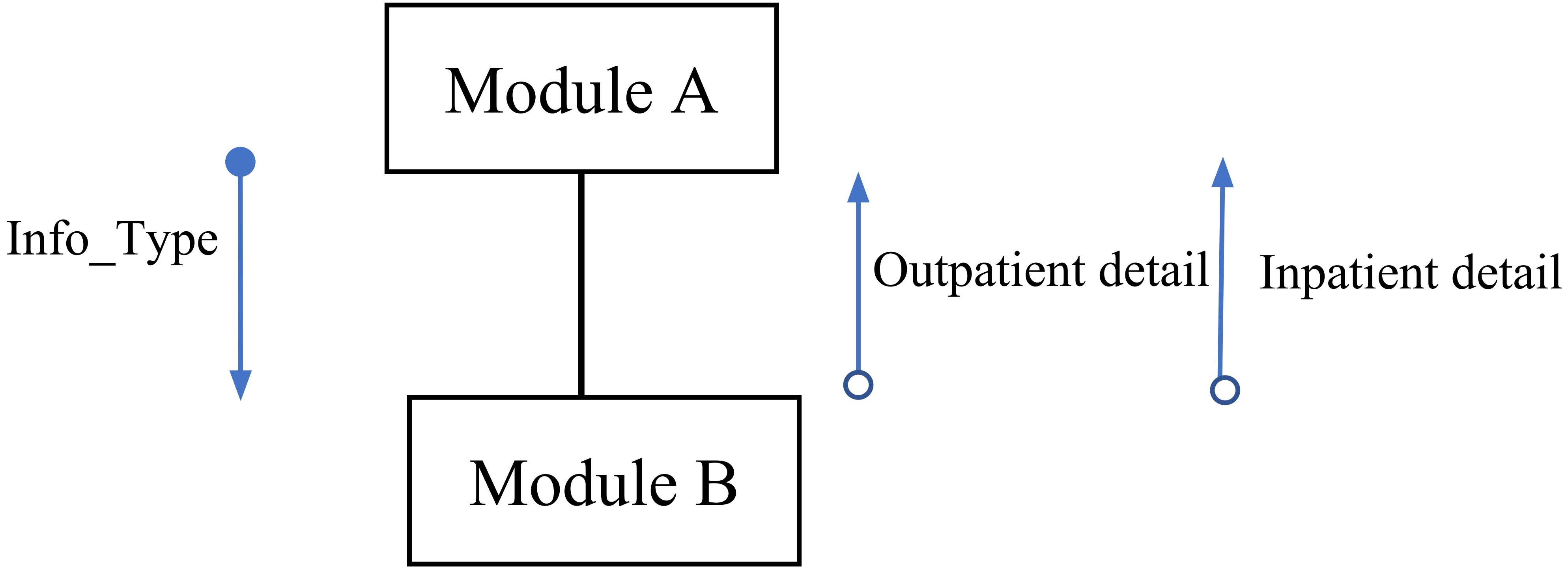
2.Stamp Coupling

- Two modules are stamp coupled if they communicate using composite data items such as structure, objects, etc.



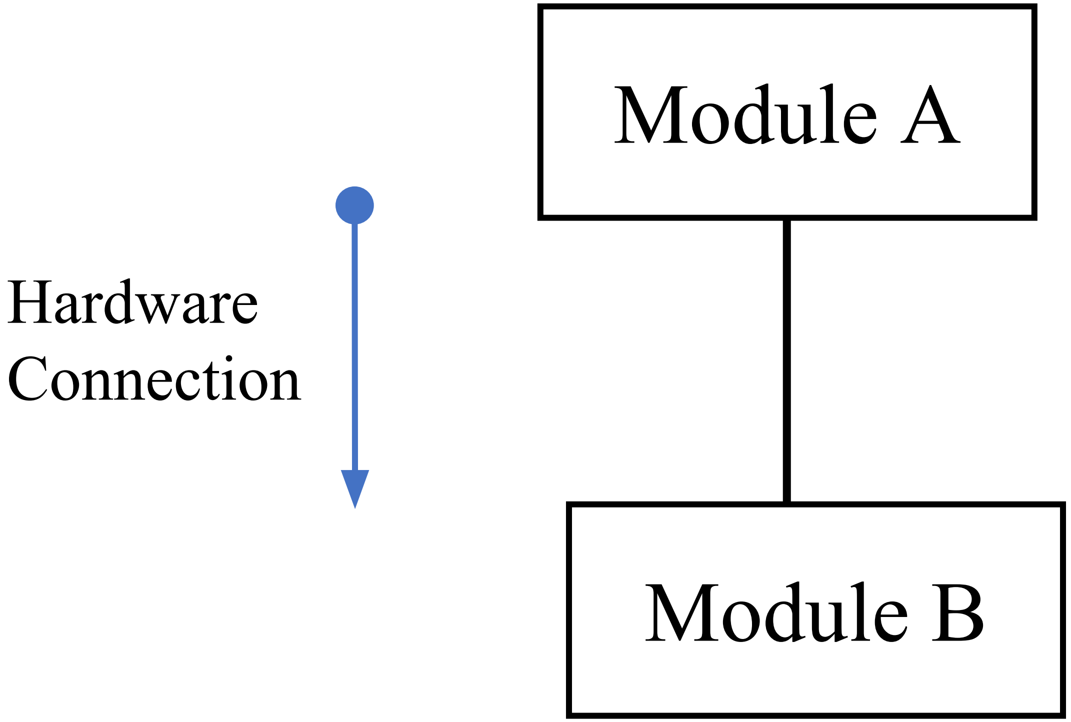
3.Control coupling

- Control Coupling exists among two modules if data from one module is used to direct the instruction execution in another.
- That means control information is passed from one module to another.



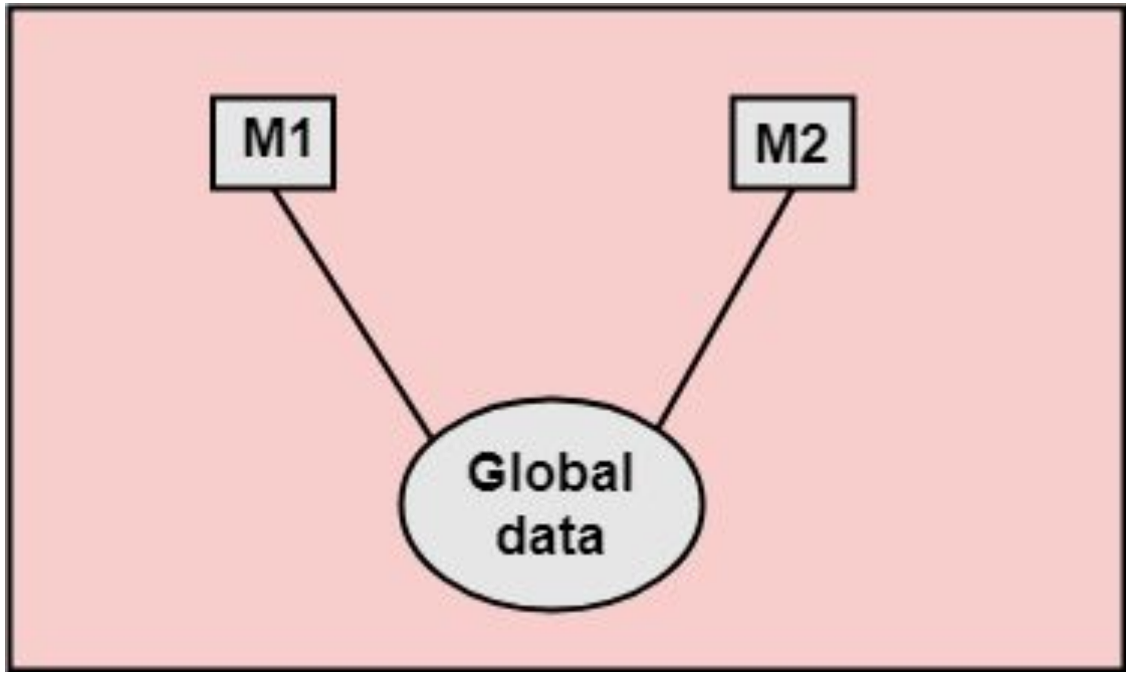
4.External coupling

- In external coupling, the modules depend on other modules, external to the software being developed or to a hardware.
- Ex- protocol, external file, device format, etc.



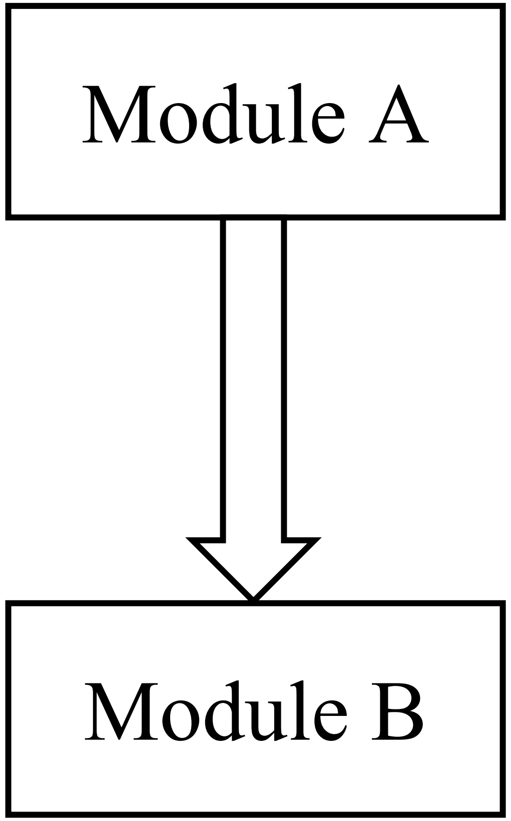
5.Common coupling

- Two modules are common coupled if they share information through some global data items or data structure or data area(processor or memory area).
-

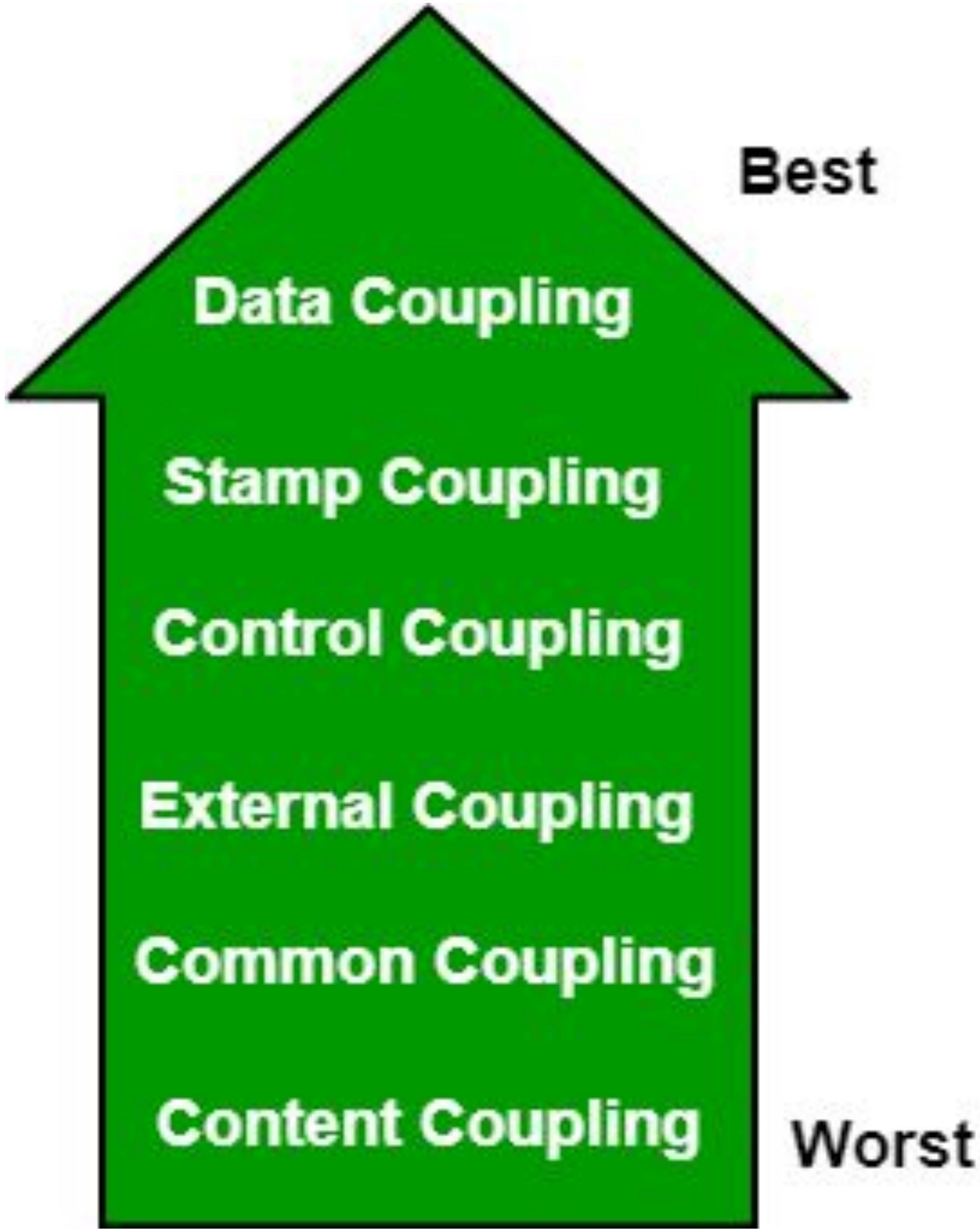


6.Content coupling

- The worst type of coupling .
- One module is directly referring to the inner working of another module.
- That means one module can alter the data in another module.



Accessing local data of another module.



Best

Data Coupling

Stamp Coupling

Control Coupling

External Coupling

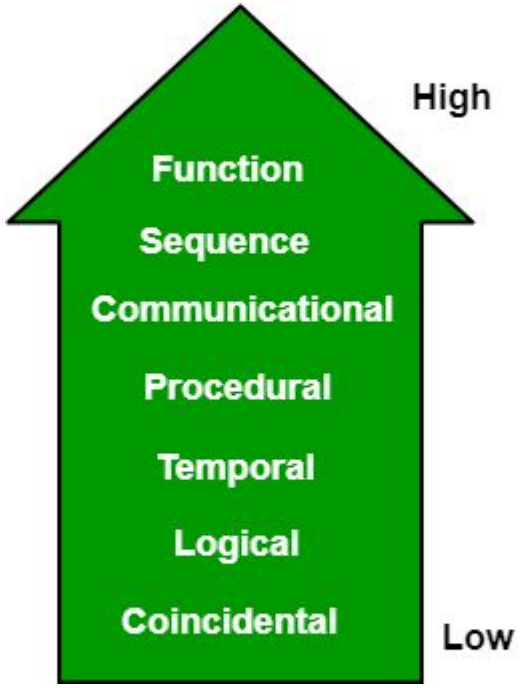
Common Coupling

Content Coupling

Worst

8.6 Cohesion

- Cohesion is a measure of the degree to which the elements of the module are functionally related.
- It is the degree to which all elements directed towards performing a single task are contained in the component.
- Basically, cohesion is the internal glue that keeps the module together.
- A good software design will have high cohesion.
- SEVEN types of cohesion are:
    1. Functional cohesion
    2. Sequential cohesion
    3. Communication cohesion
    4. Procedural cohesion
    5. Temporal cohesion
    6. Logical cohesion
    7. Coincidental cohesion



High

Function

Sequence

Communicational

Procedural

Temporal

Logical

Coincidental

Low

1. Functional Cohesion

- The most desirable type of cohesion.
- All the elements or units within a module are based on one function.
- Example: maintain temperature for steel furnace, calculate interest rate, sort the array ,etc.

2.Sequential Cohesion

- An element outputs some data that becomes the input for other element in a module, i.e., data flow between the parts.
- Example : Searching array.

3.Communication Cohesion

- Two elements operate on the same input data or contribute towards the same output data.
- Example- update record int the database and send it to the printer.

4.Procedural Cohesion

- Elements of procedural cohesion ensure the order of execution.
- Actions are still weakly connected and unlikely to be reusable.
- Example: Calculate student GPA, print student record, calculate cumulative GPA, print cumulative GPA.
-

5.Temporal Cohesion

- The elements are related by their timing involved.
- A module connected with temporal cohesion all the tasks must be executed in the same time-span.
- This cohesion contains the code for initializing all the parts of the system.
- Lots of different activities occur, all at in time.

6.Logical Cohesion

- The elements are logically related and not functionally.
- Example: A component reads inputs from tape, disk, and network.
- All the code for these functions is in the same component.
- Operations are related, but the functions are significantly different.

7.Coincidental Cohesion

- The elements are not related(unrelated).
- The elements have no conceptual relationship other than location in source code.
- It is accidental and the worst form of cohesion.
- Example: Print next line and reverse the characters of a string in a single component.

Compare Coupling and Cohesion(assignment)