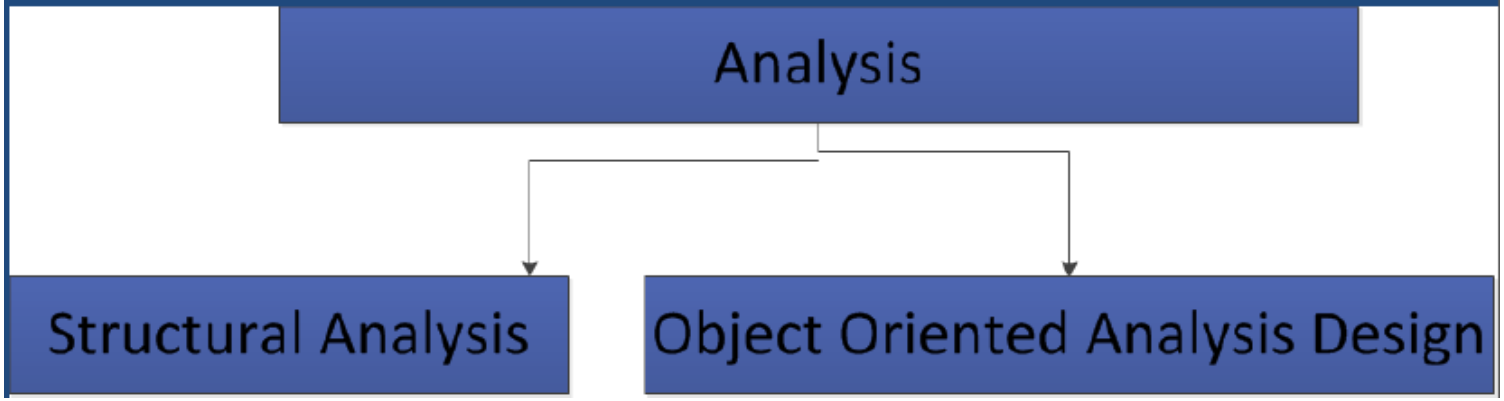
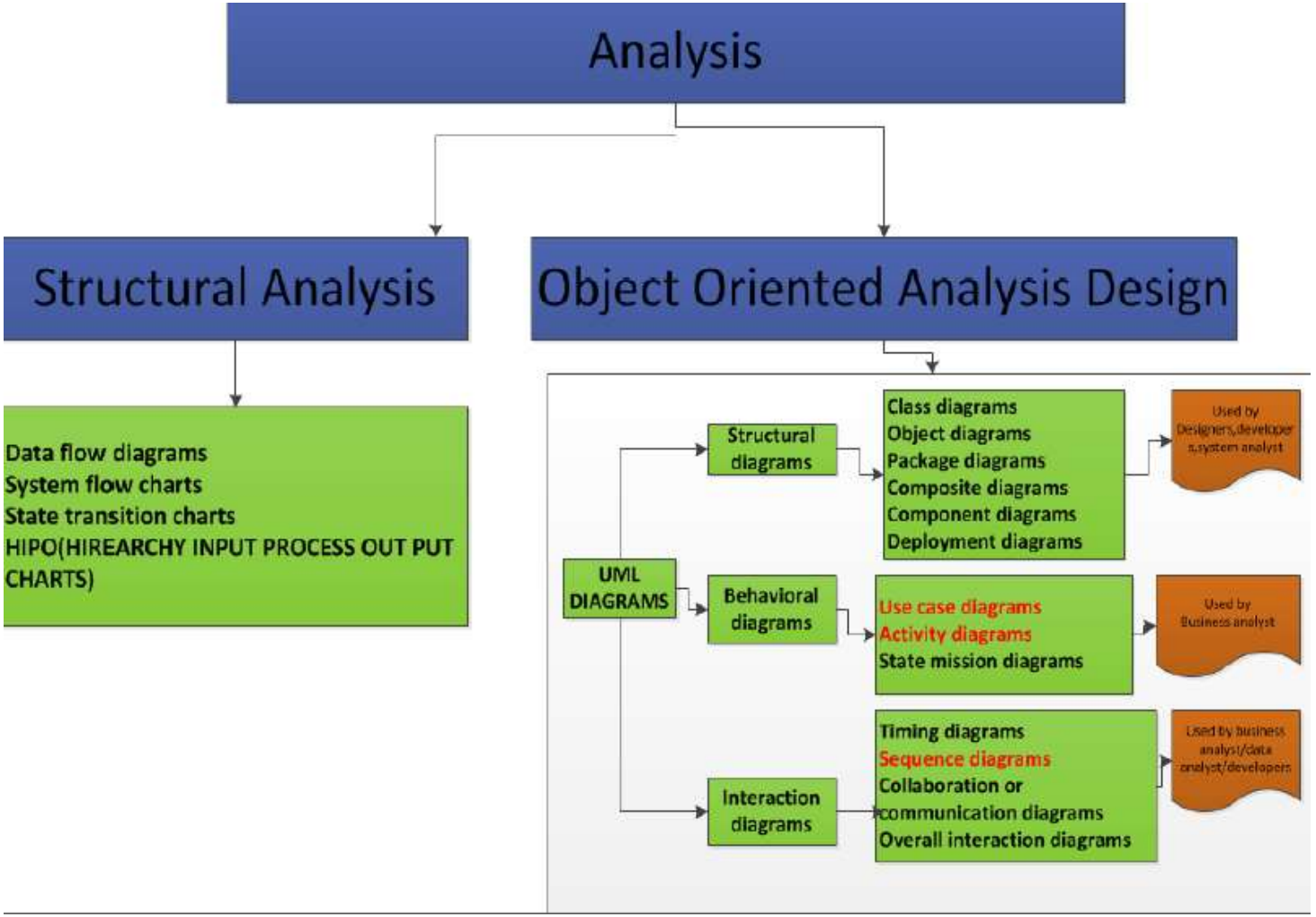
ANALYSIS

ANALYSIS

Business analysis is the discipline of identifying business needs and determining solutions to business problems. Solutions often include a systems development component, but may also consist of process improvement, organizational change or strategic planning and policy development. The person who carries out this task is called a business analyst or BA.[2]

Business analysts who work solely on developing software systems may be called IT business analysts, technical business analysts, online business analysts, business systems analysts, or systems analysts.




Analysis
Structural Analysis
Object Oriented Analysis Design
Data flow diagrams
System flow charts
State transition charts
HIPO(HIREARCHY INPUT PROCESS OUT PUT CHARTS)
UML DIAGRAMS
Structural diagrams
Behavioral diagrams
Interaction diagrams
Class diagrams
Object diagrams
Package diagrams
Composite diagrams
Component diagrams
Deployment diagrams
Use case diagrams
Activity diagrams
State mission diagrams
Timing diagrams
Sequence diagrams
Collaboration or communication diagrams
Overall interaction diagrams
Used by Designers,developer s,system analyst
Used by Business analyst
Used by business analyst/data analyst/developers

Structured analysis

•A widely-used top-down method for defining system inputs, processes and outputs.

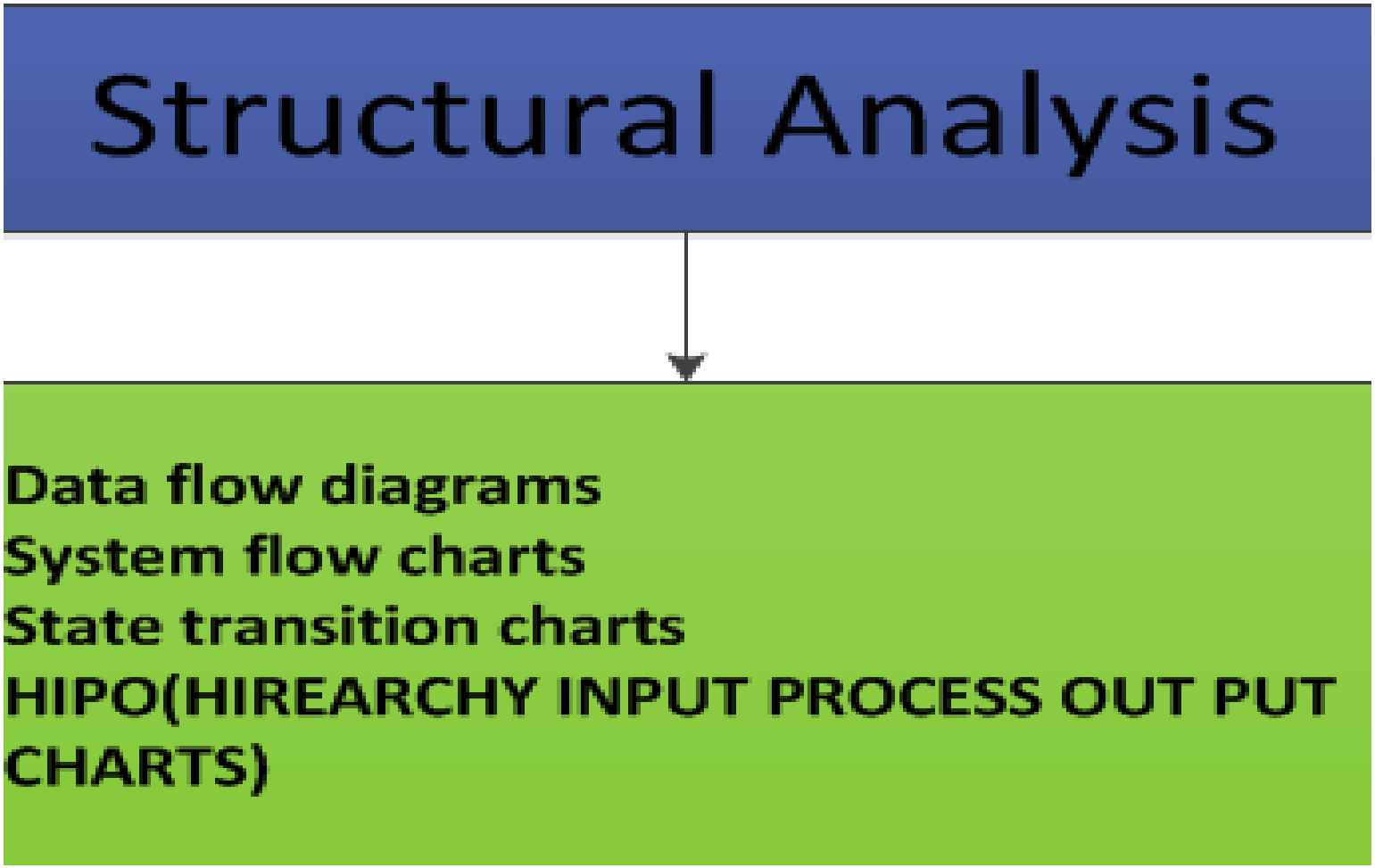
•It shows how information flows through a system, using several diagrams showing progressively more and more detail at each level.

•The primary tool of structured analysis is the Data Flow Diagram (DFD).

Structural analysis

•Examination of the different components or elements that make up an organization or system, to discover their interrelationships and relative importance in the realization of its goals or purpose.

Structural analysis



Data Flow Diagrams

A structured analysis technique that employs a set of visual representations of the data that moves through the organization, the paths through which the data moves, and the processes that produce, use, and transform data.

Why Data Flow Diagrams?

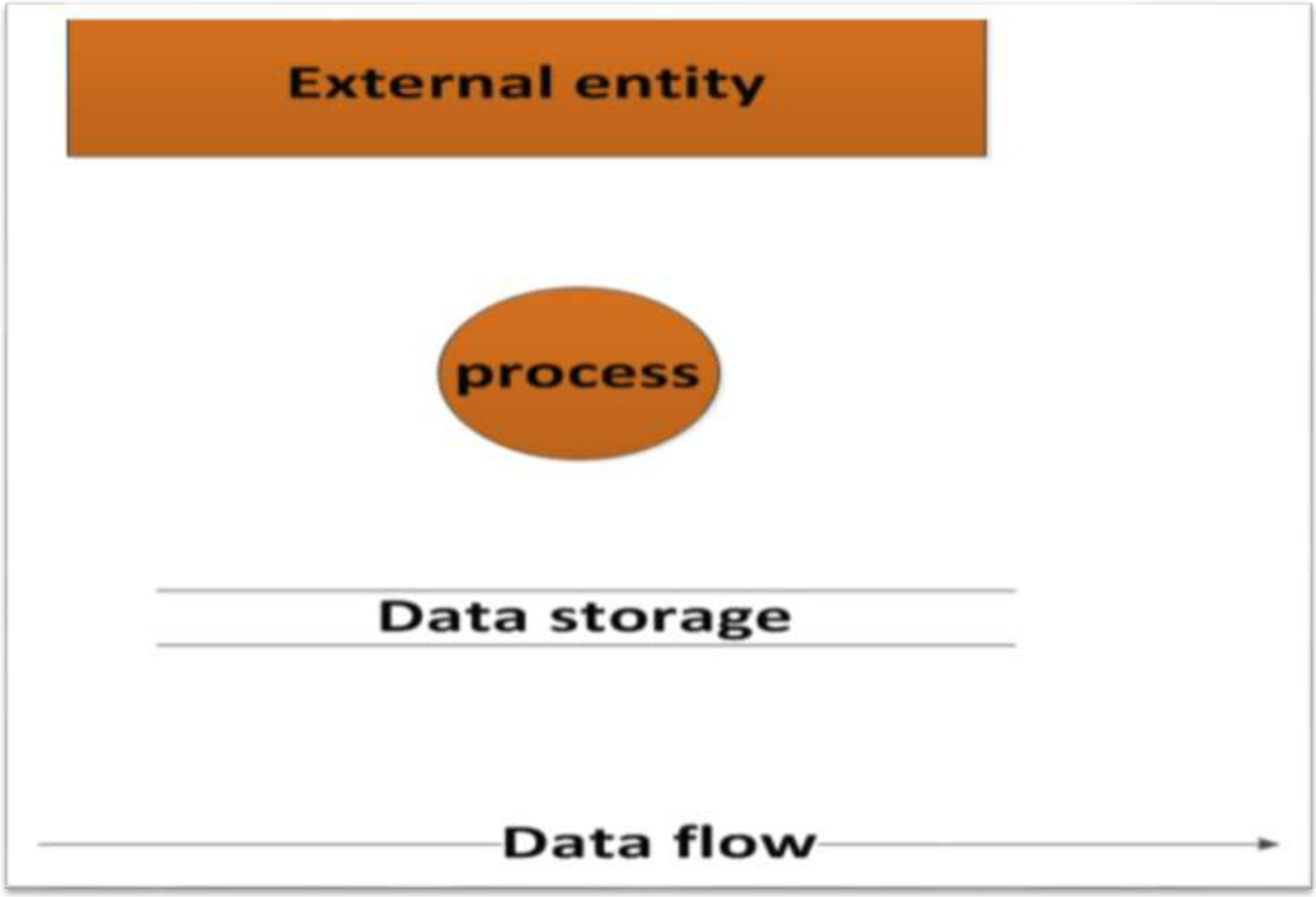
- Can diagram the **organization** or the **system**
- Can diagram the **current** or **proposed** situation
- Can facilitate **analysis** or **design**
- Provides a good bridge from analysis to design
- Facilitates communication with the user at all stages

Data flow diagrams

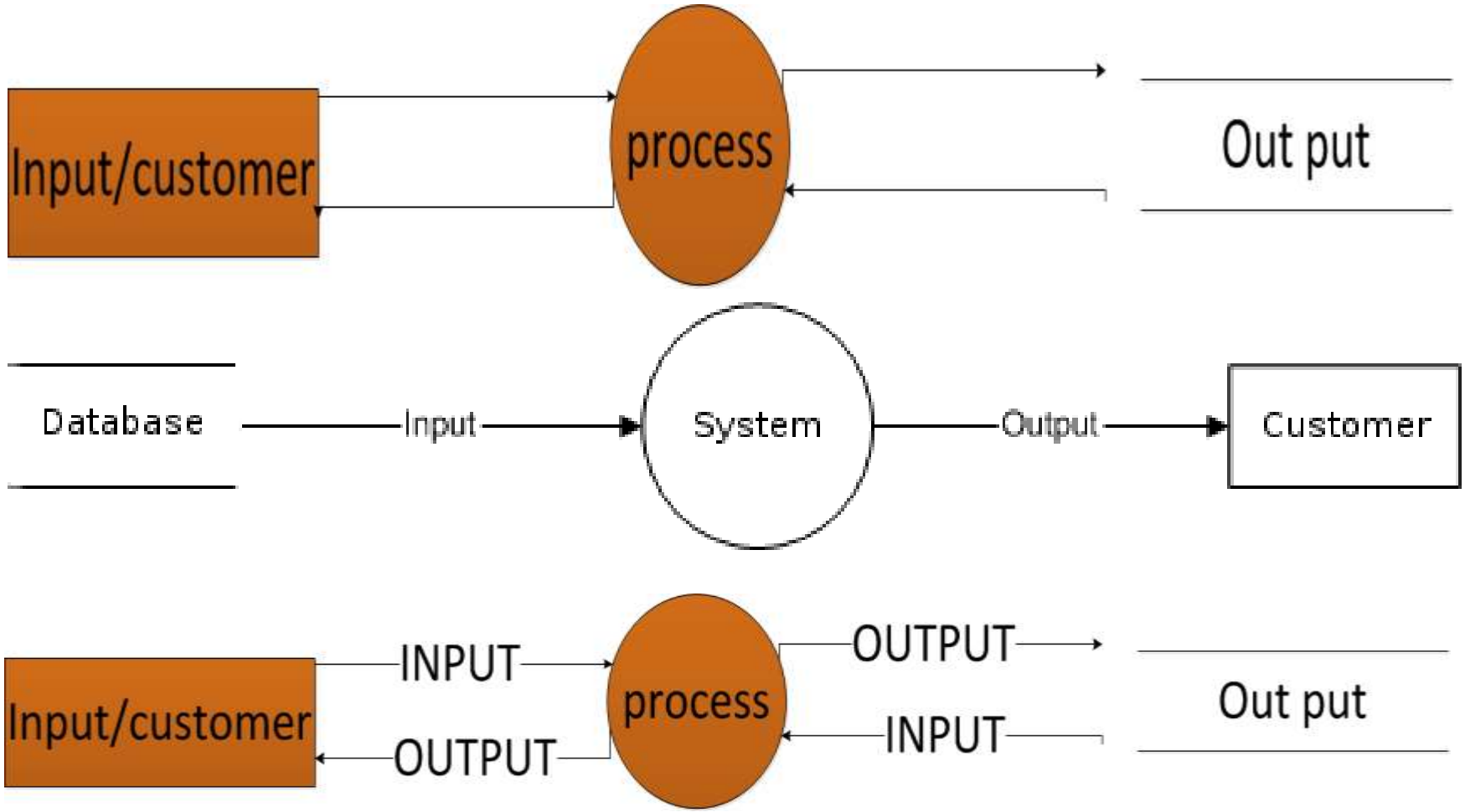
•A **data flow diagram** (DFD) is a graphical representation of the "flow" of data through an information system, modeling its *process* aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. [2]DFDs can also be used for the visualization of data processing (structured design).

•A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

DFD Notations



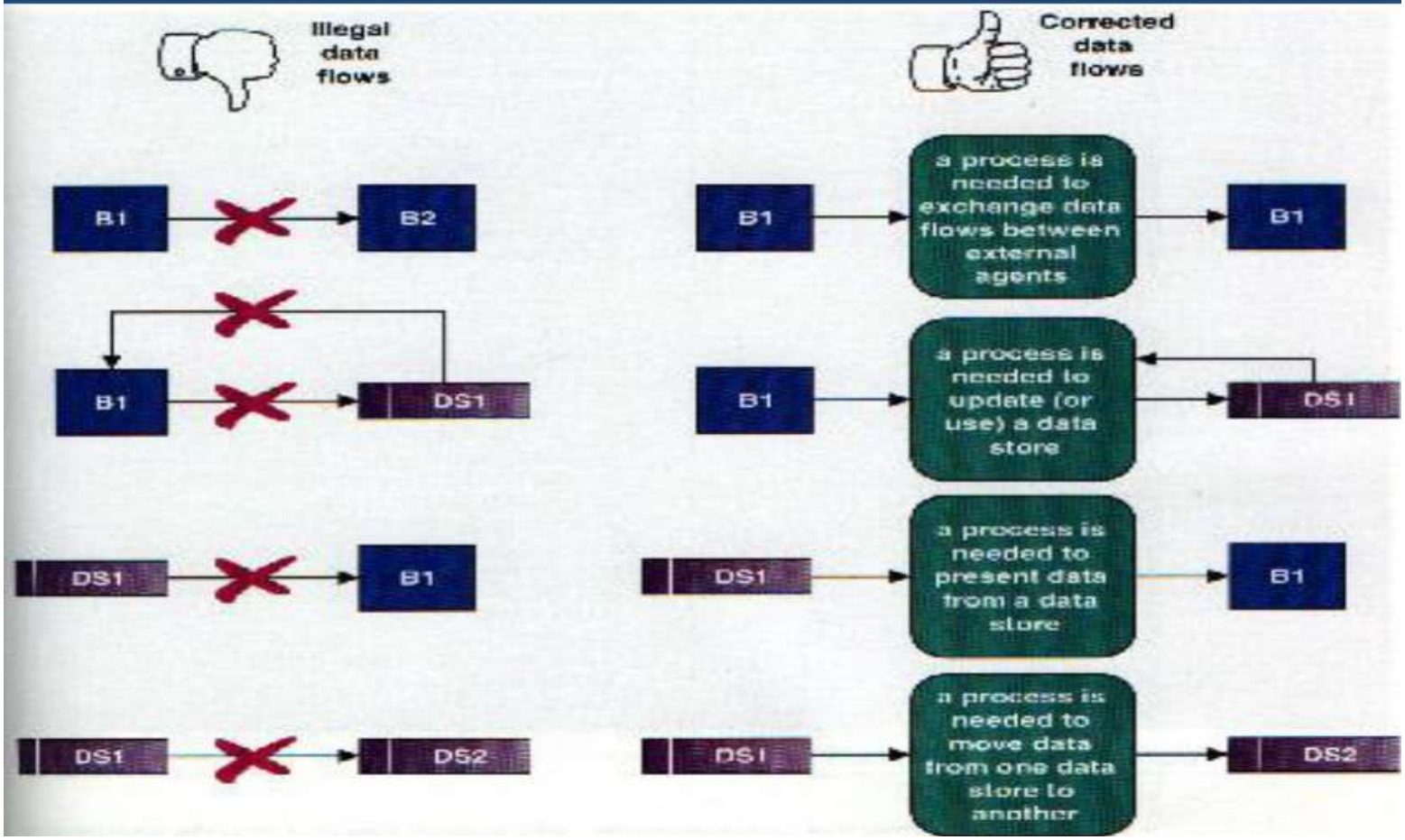
Data Flow Diagrams



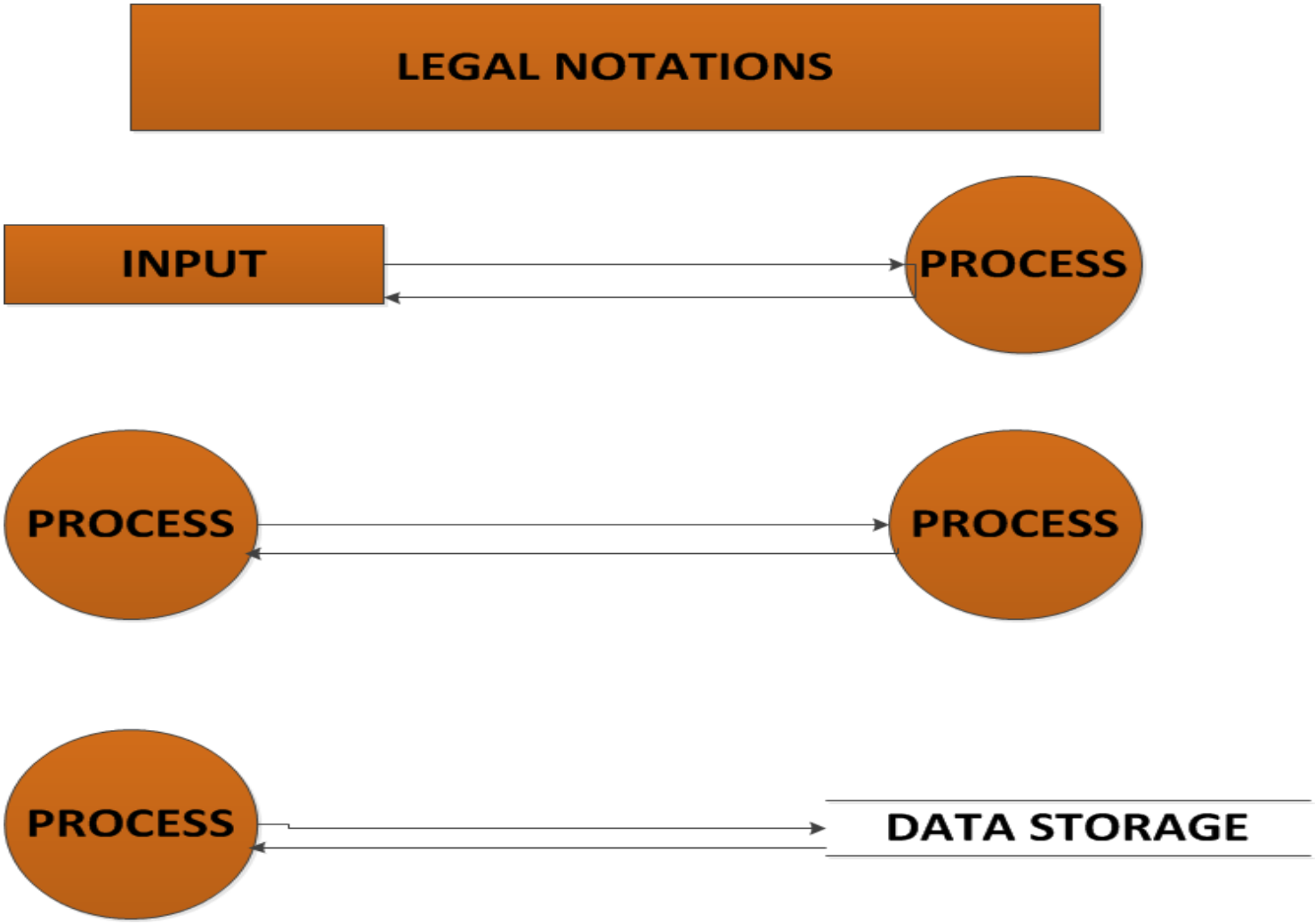
RULES FOR USING DFD

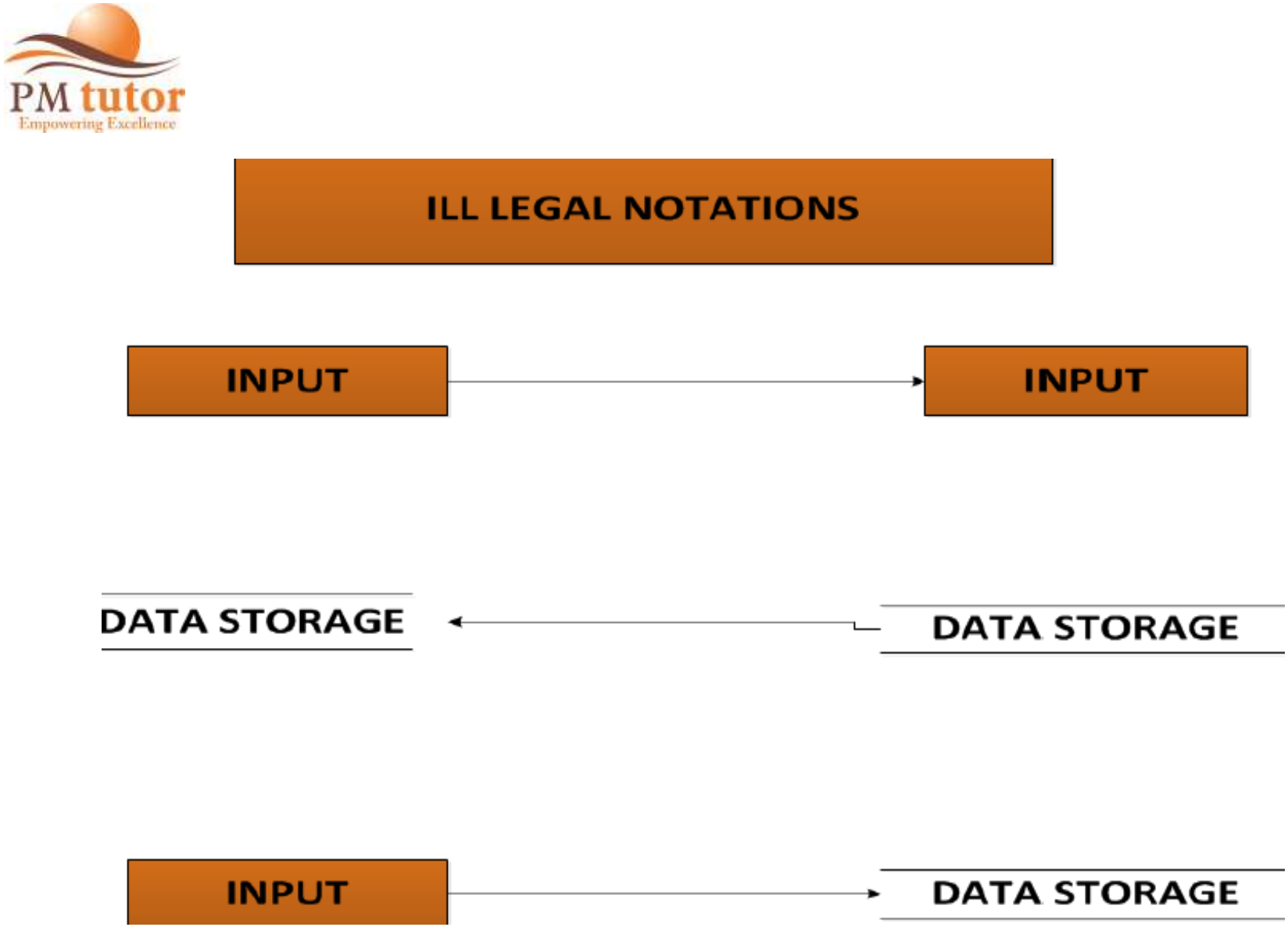
- LEGAL RULES NOTATIONS
- ILLEGAL RULES NOTATIONS

Illegal Data Flows



Legal Notations




ILL LEGAL NOTATIONS
INPUT
INPUT
DATA STORAGE
DATA STORAGE
INPUT
DATA STORAGE

DFD'S

•The dfd is essential tool for creating formal descriptions of business process and data flows

Advantages of the Data Flow Diagram Approach

- **Four advantages over narrative explanations of data movement**

–Freedom from committing to the technical implementation too early

–Understanding of the interrelationships of systems and subsystems

–Communicating current system knowledge to users

–Analysis of the proposed system

Data flow diagram symbols

- **Four basic symbols are**

–A double square for an external entity--a source or destination of data

–An arrow for movement of data from one point to another

–A rectangle with rounded corners for the occurrence of transforming process

–An open-ended rectangle for a data store

External Entities

•Represent people or organizations outside of the system being studied.

•Shows the initial source and final recipient of data and information.

•Should be named with a noun, describing that entity

- External entities may be

–A person, such as **CUSTOMER** or **STUDENT**

–A company or organization, such as **BANK** or **SUPPLIER**

–Another department within the company, such as **ORDER FULFILLMENT**

–Another system or subsystem, such as the **INVENTORY CONTROL SYSTEM**

Processes

- **Represent either:**
  - A whole system
  - A subsystem
  - Work being done, an activity

- **Names should be in the form verb-adjective-noun**
  - The exception is a process that represents an entire system or subsystem

Data Stores

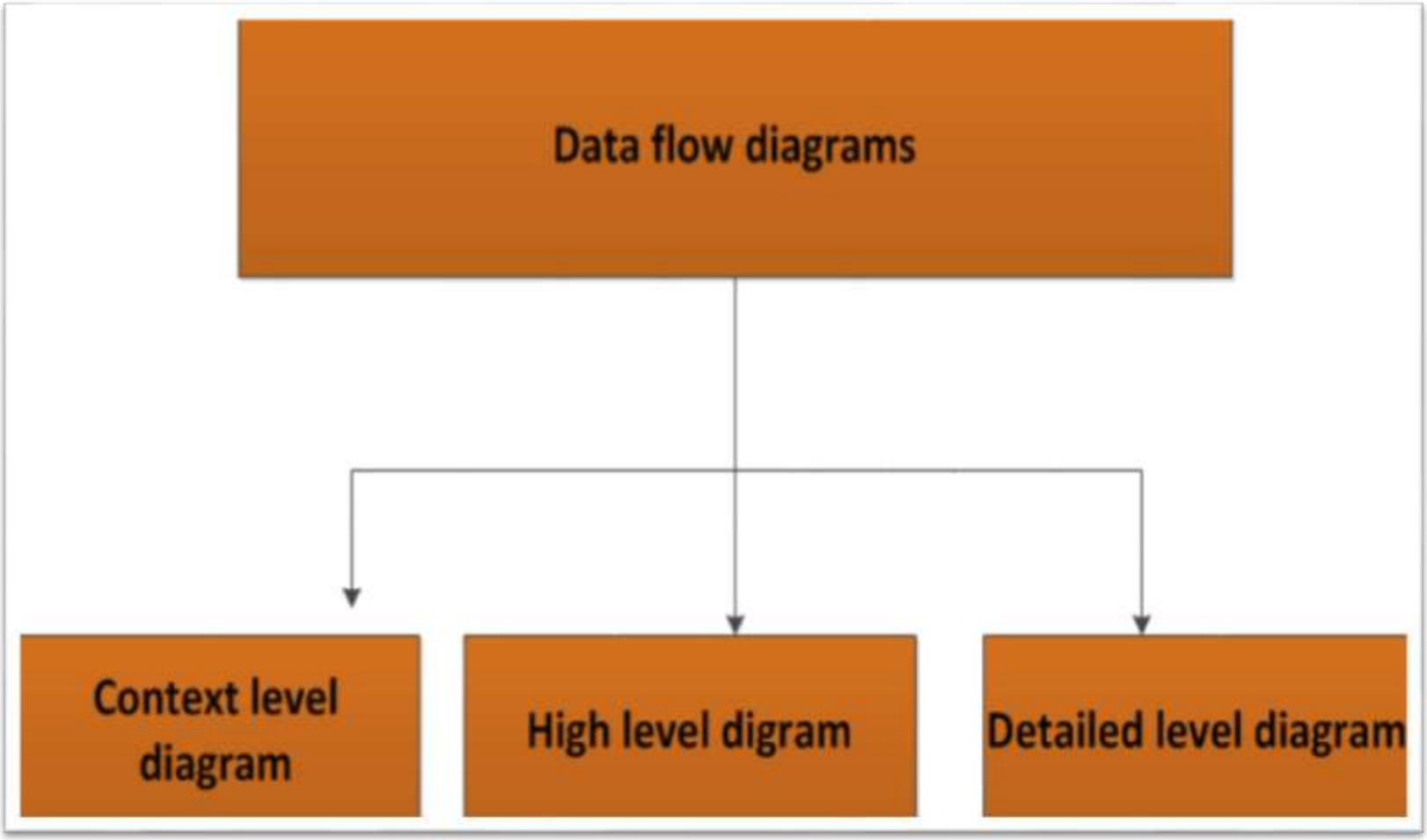
- Name with a noun, describing the data
- **Data stores are usually given a unique reference number, such as D1, D2, & D3**
- Include any data stored, such as:
  - A computer file or database
  - A transaction file
  - A set of tables
  - A manual file of records

D1 Customer Master

Data Flow

- Shows the data about a person, place, or thing that moves through the system
- Names should be a noun that describes the data moving through the system
- Arrowhead indicates the flow direction
- Use double headed-arrows only when a process is reading data and updating the data on the same table or file

DFD Diagram


Data flow diagrams
Context level diagram
High level digram
Detailed level diagram

Context Level Diagram

- Just one process
- All sources and sinks that provide data to or receive data from the process
- Major data flows between the process and all sources/sinks
- No data stores

Context Level DFD'S

- SHOWS INTERACTION BETWEEN EXTERNAL ENTITY TO MAIN SYSTEM OR PROCESS FOR THE PURPOSE OF CONSUMING AND PRODUCING DATA
- EXAMPLE
- ONLINE RAILWAY RESERVATION SYSTEM

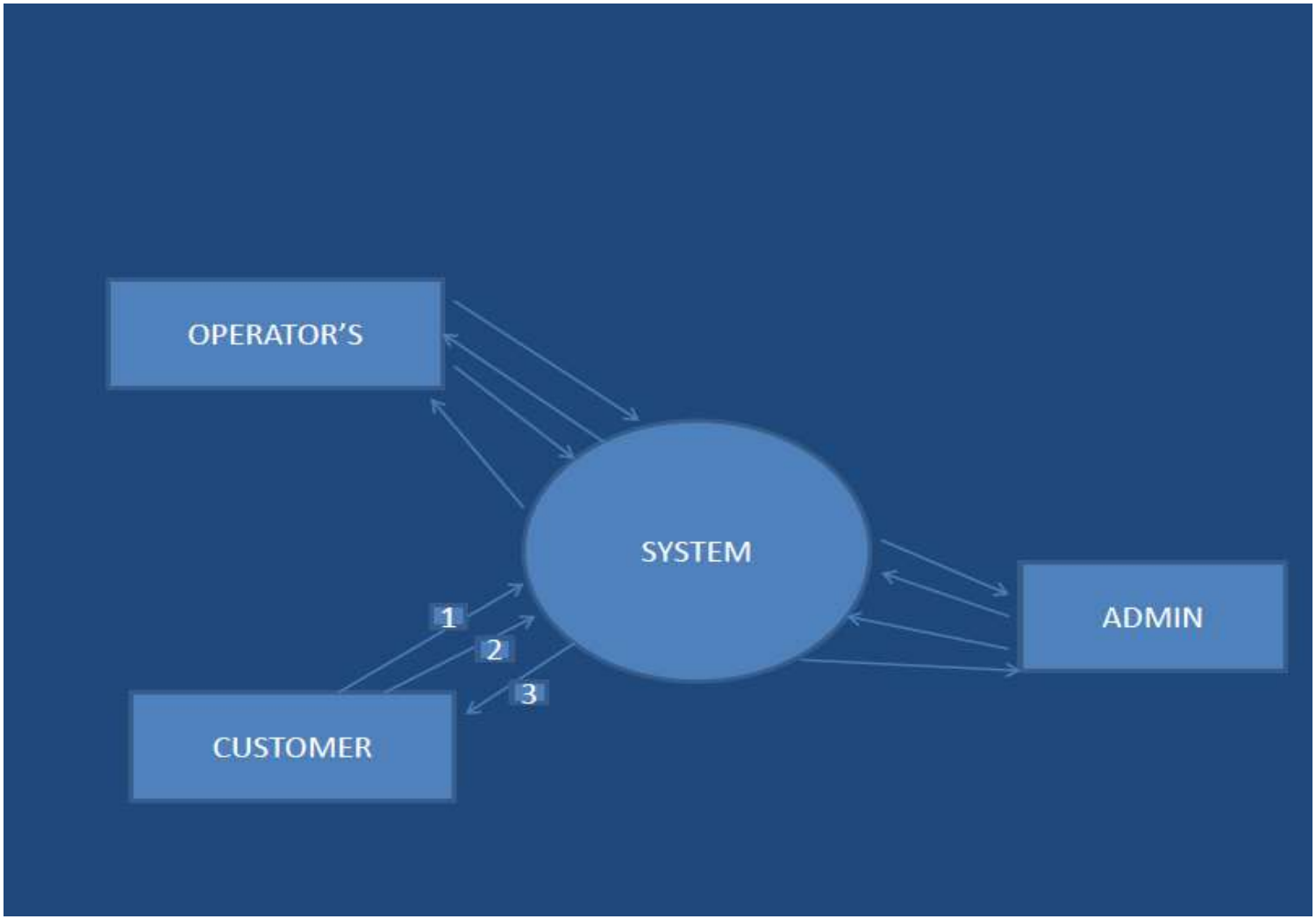
Context Level Data Flow Diagram

•Contains only one process, representing the entire system

•The process is given the number zero

•Allexternalentitiesareshownonthecontextdiagramaswellasmajordataflowtoand fromthem

•The diagram does not contain any data stores


OPERATOR'S
SYSTEM
ADMIN
1
2
3
CUSTOMER

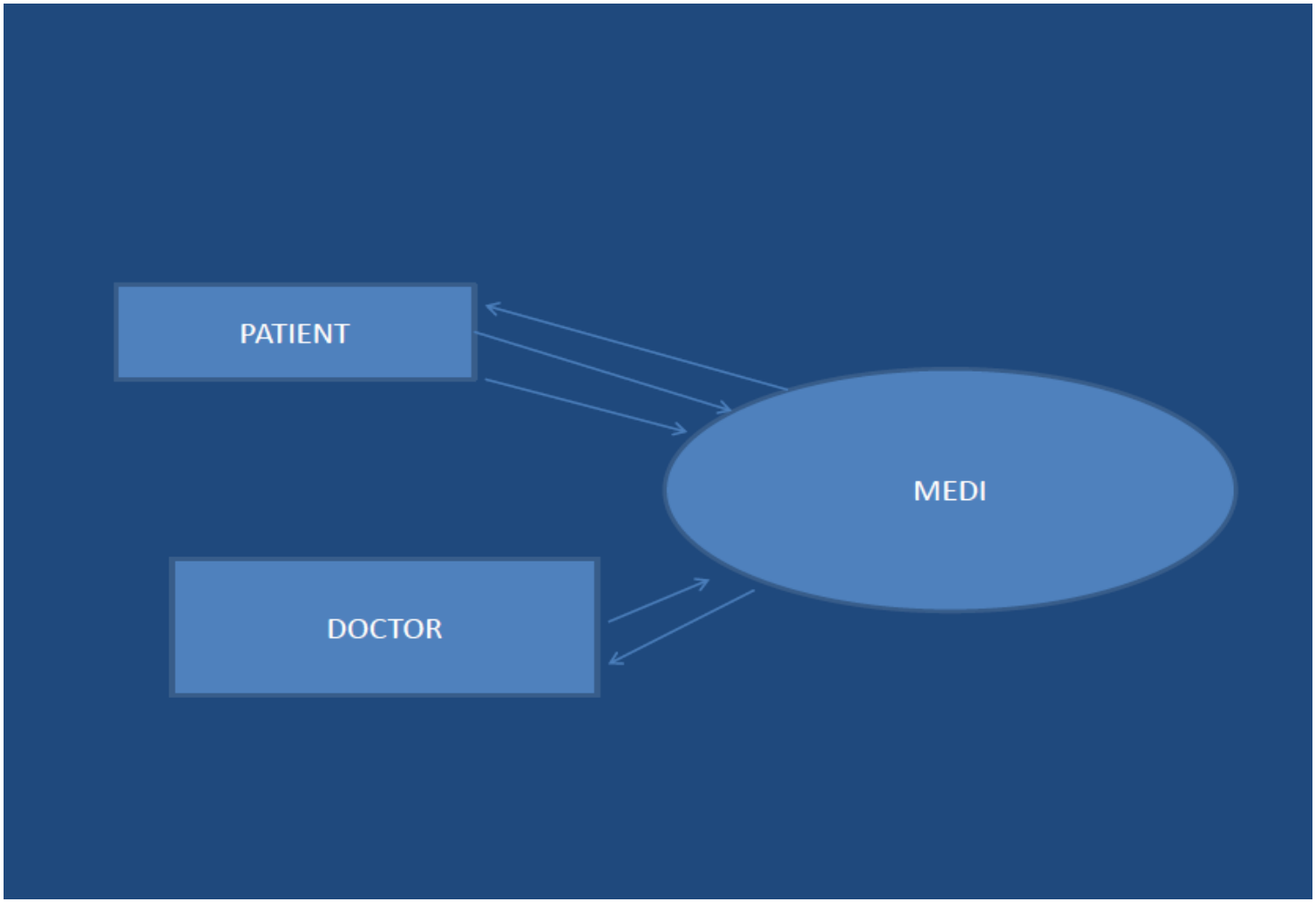
TOP-LEVEL DFD

- Diagram is the explosion of the context level diagram

- Should include up to7 or 9 processes
  - –Anymore will result in a clustered diagram

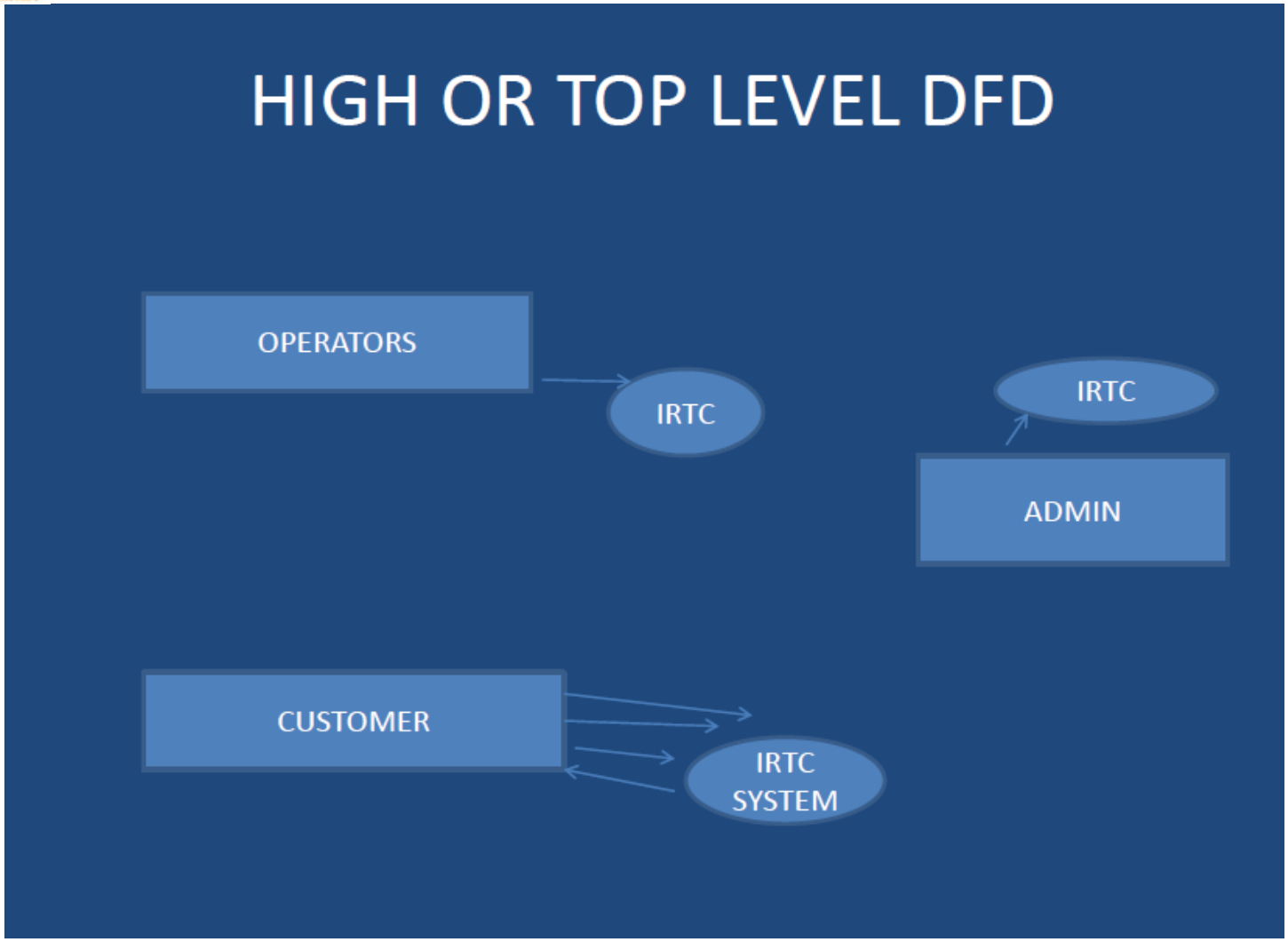
- Processes are numbered with an integer

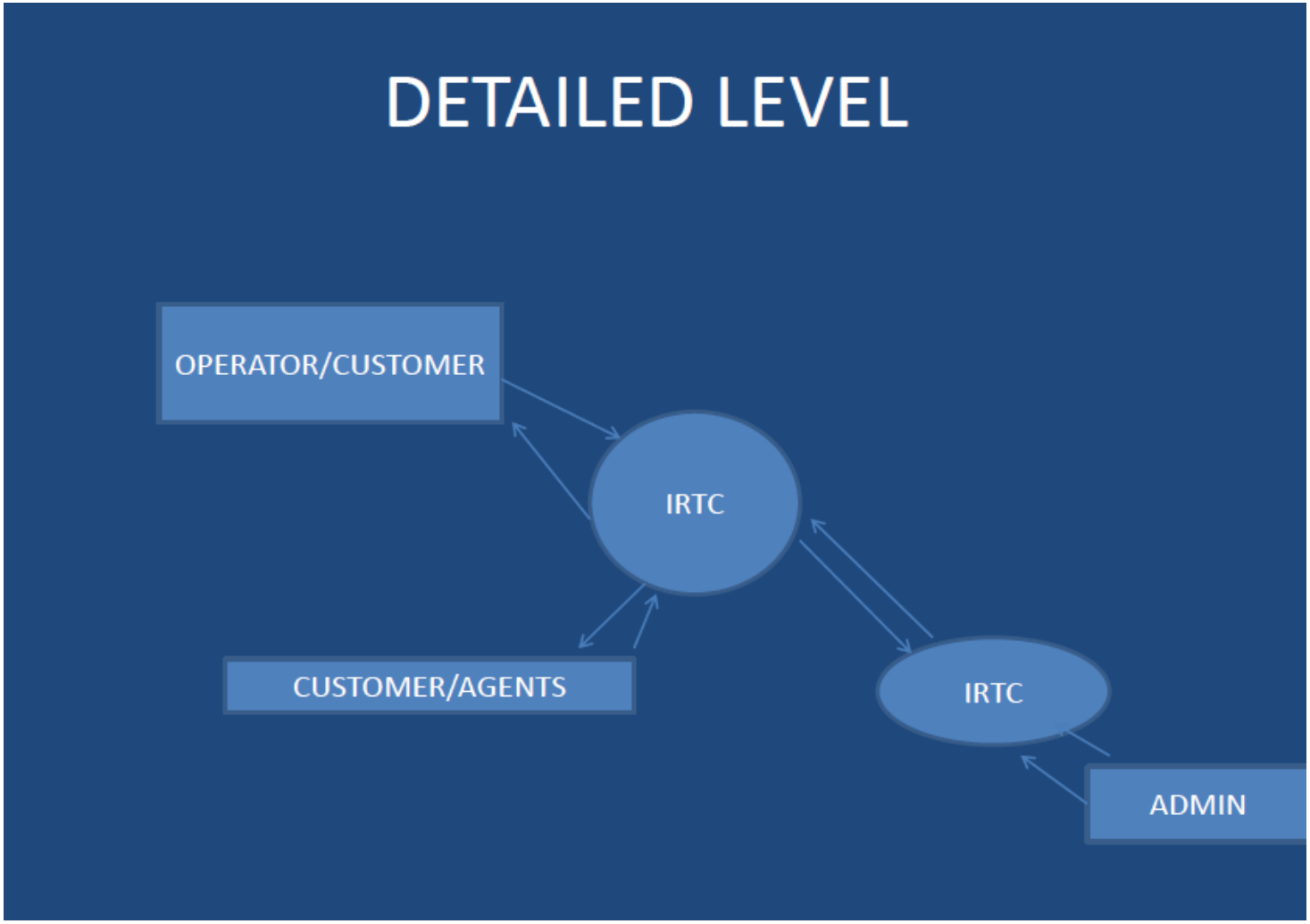
- The major data store sandal external entities are included on Diagram


PATIENT
MEDI
DOCTOR

- 1 LOGIN

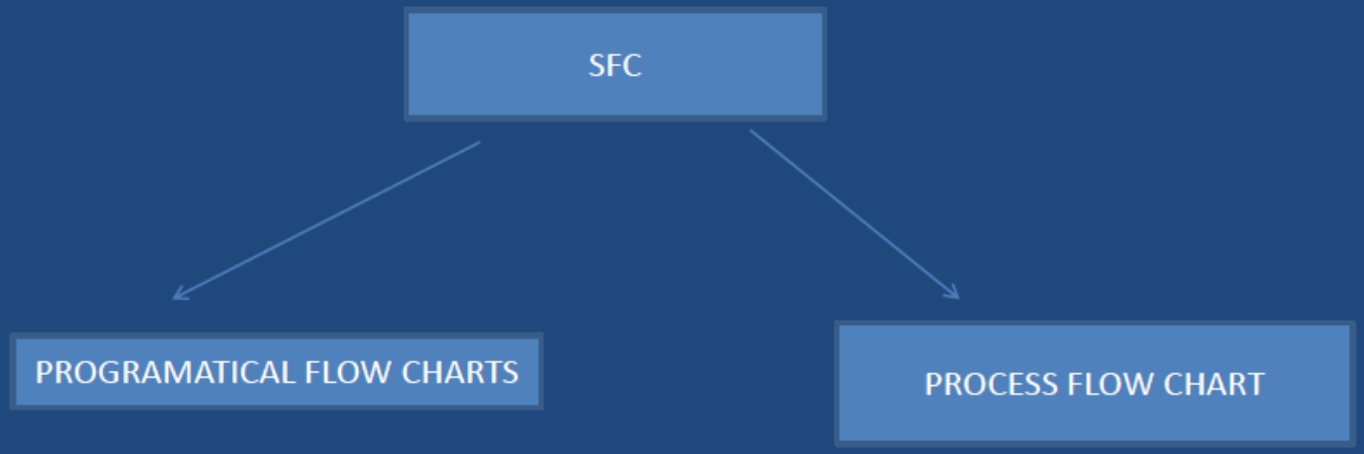
- 2 USERNAME


HIGH OR TOP LEVEL DFD
OPERATORS
IRTC
IRTC
ADMIN
CUSTOMER
IRTC
SYSTEM


DETAILED LEVEL
OPERATOR/CUSTOMER
IRTC
CUSTOMER/AGENTS
IRTC
ADMIN

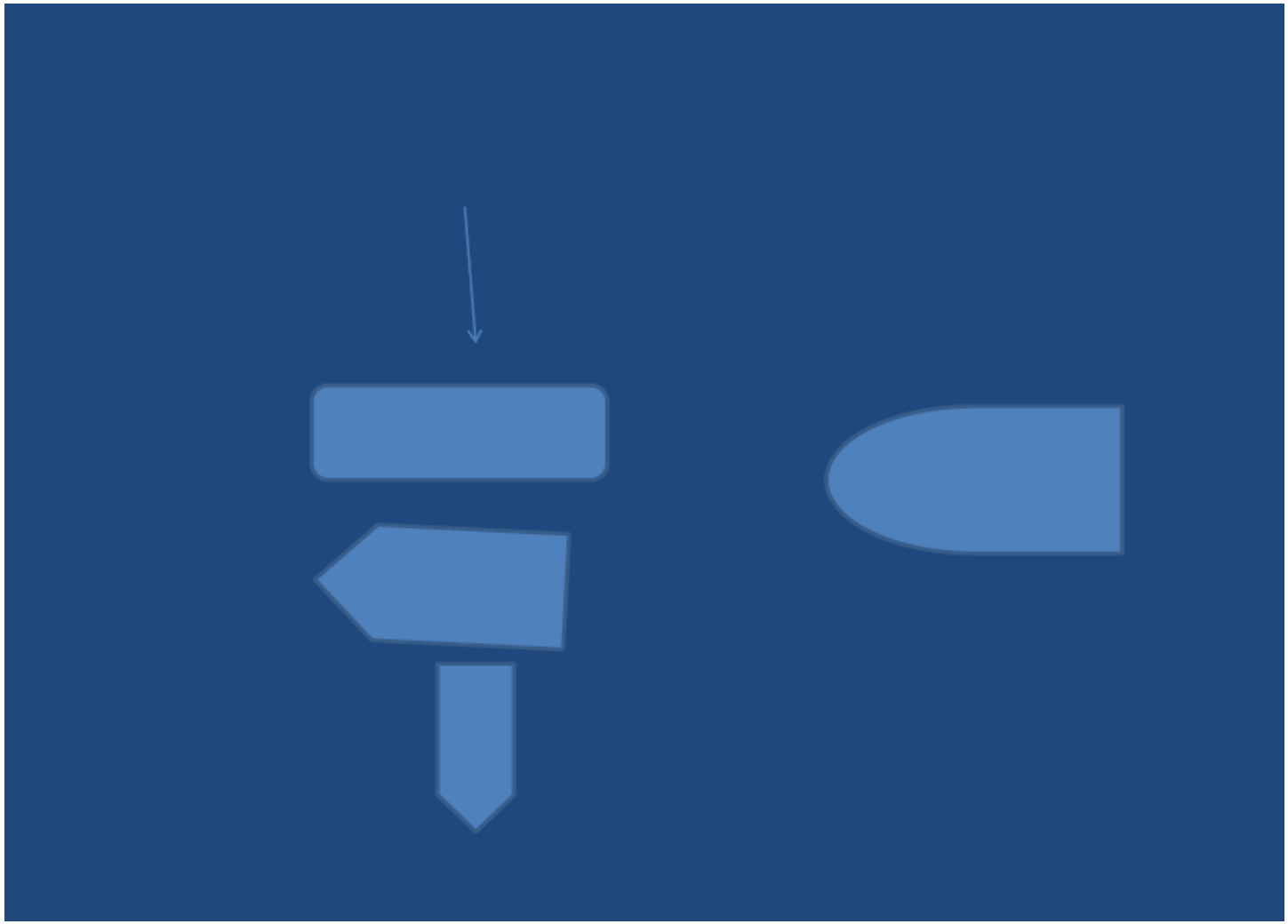
SYSTEM FLOW CHARTS

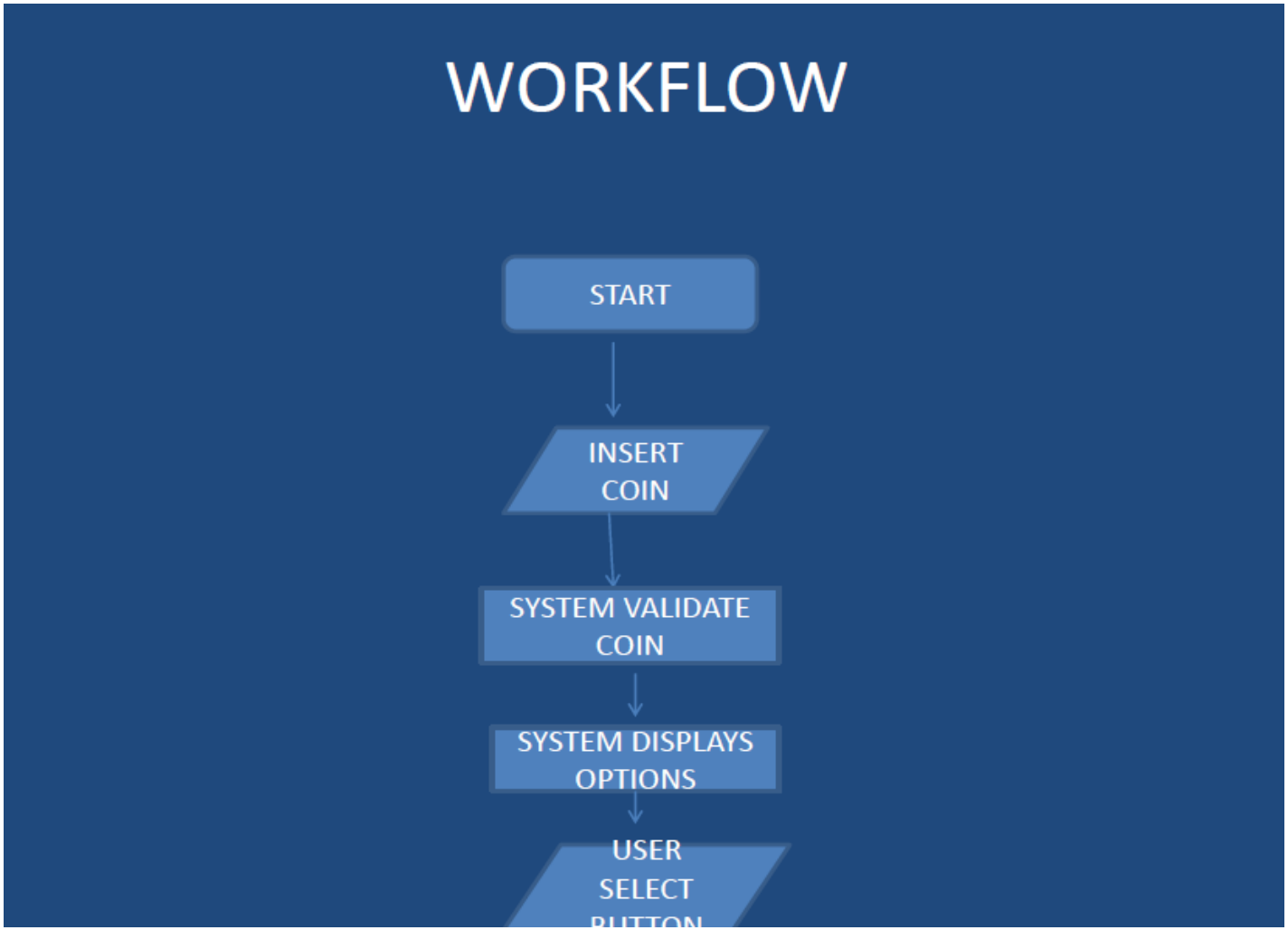
- CONTROL FLOW OF THE SYSTEM OR DIRECTION OF BUSINESS NEEDS OR FLOW OF DATA WITH SYSTEM

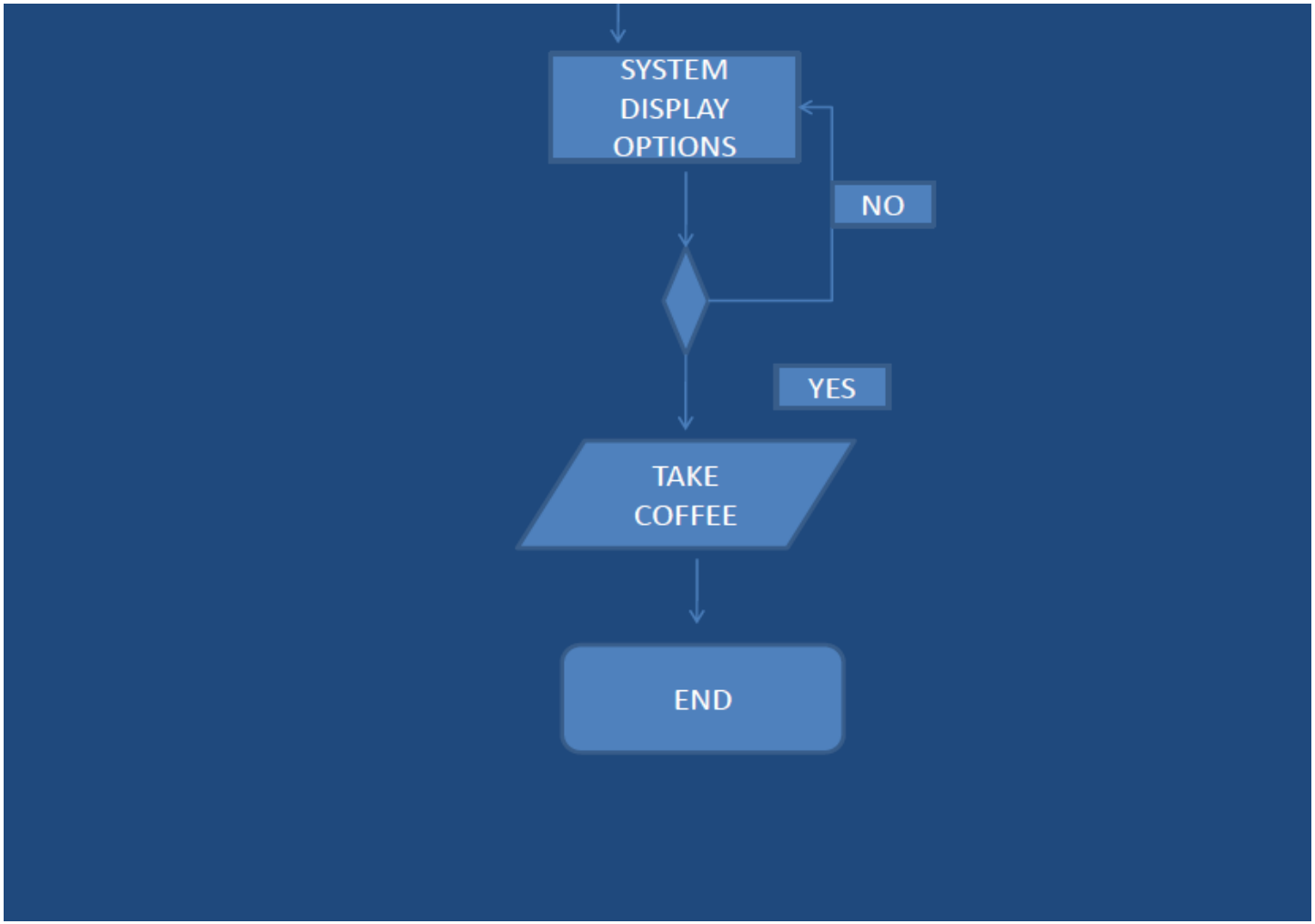


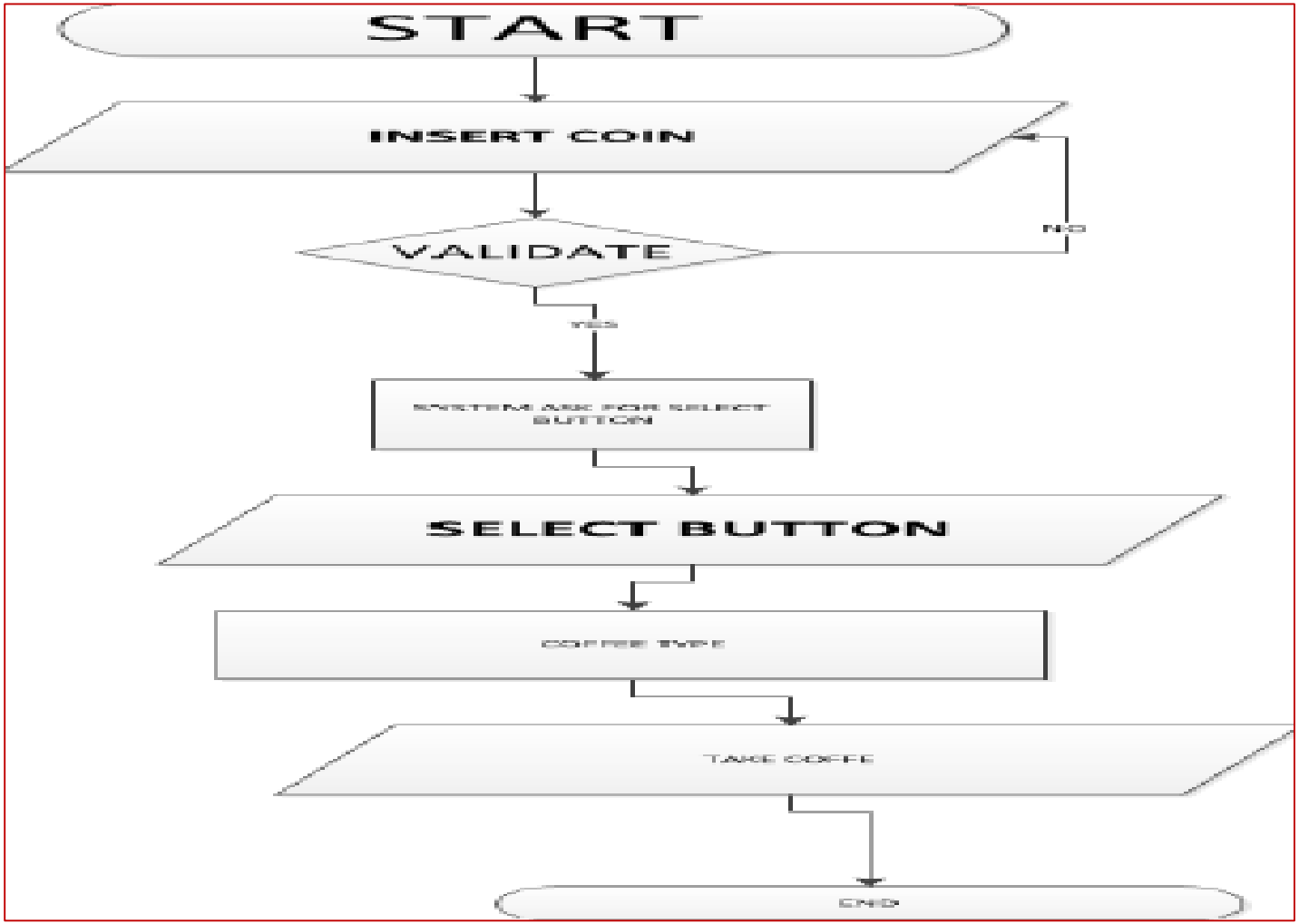
PROCESS FLOW CHARTS

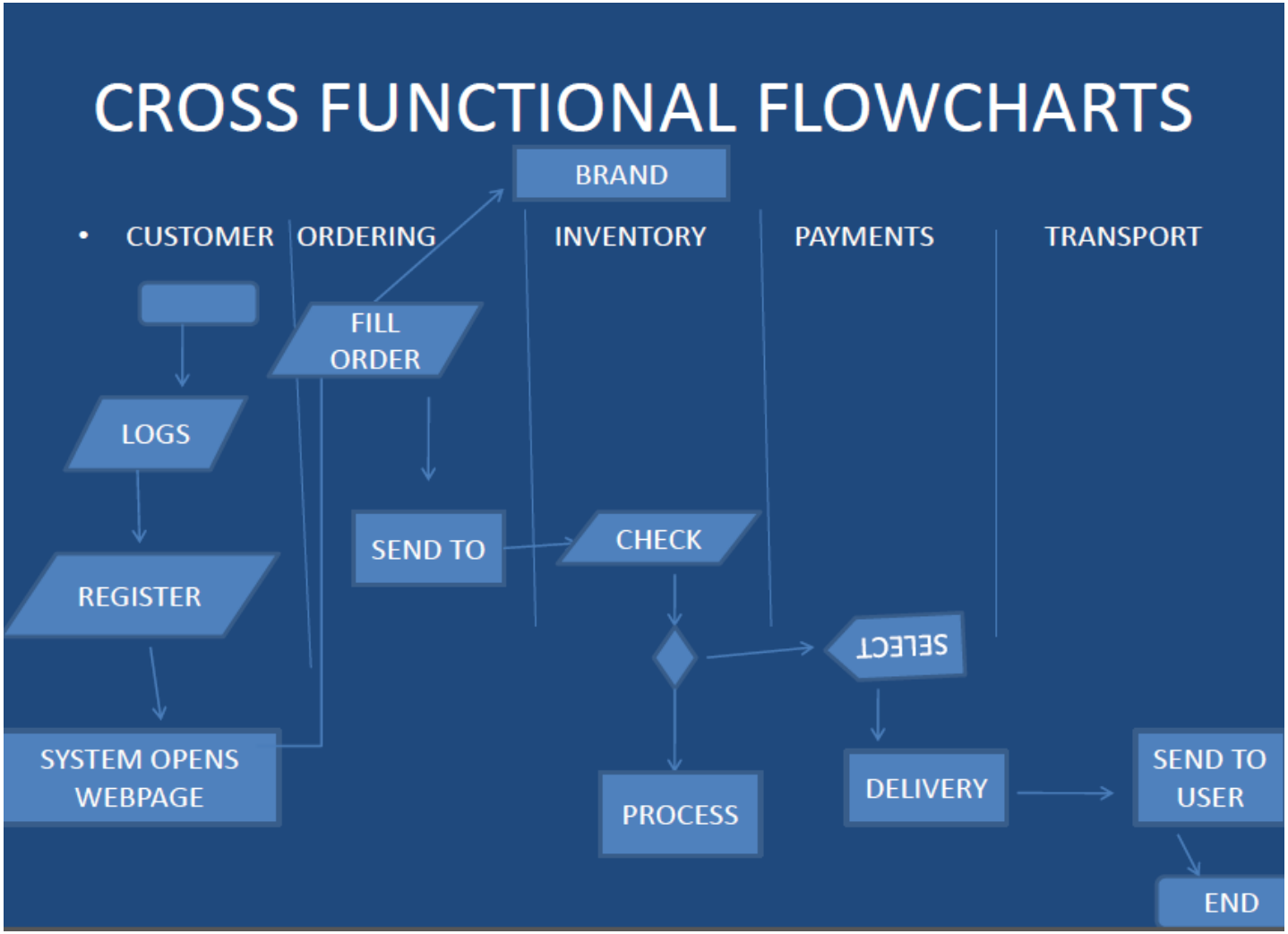
- BASIC FLOW CHART/WORKFLOW
- CROSS FUNCTIONAL FLOWCHARTS
- NOTATION;

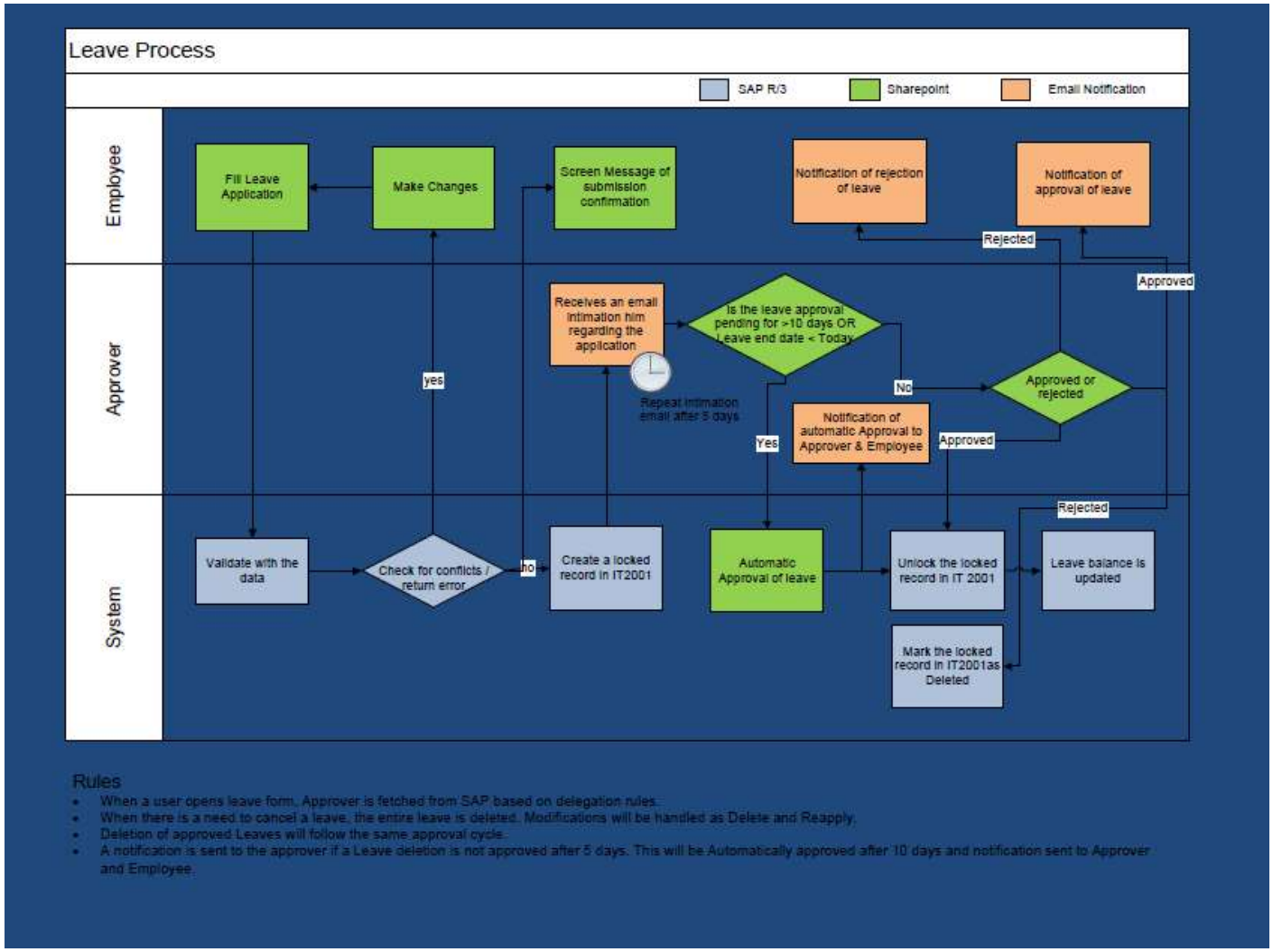



WORKFLOW
START
INSERT COIN
SYSTEM VALIDATE COIN
SYSTEM DISPLAYS OPTIONS
USER SELECT


SYSTEM
DISPLAY
OPTIONS
NO
YES
TAKE
COFFEE
END


START
INSERT COIN
VALIDATE
NO
YES
SYSTEM ASK FOR SELECT BUTTON
SELECT BUTTON
COFFEE TYPE
TAKE COFFE
END


CROSS FUNCTIONAL FLOWCHARTS
BRAND
CUSTOMER
ORDERING
INVENTORY
PAYMENTS
TRANSPORT
FILL ORDER
LOGS
REGISTER
SEND TO
CHECK
SELECT
SYSTEM OPENS WEBPAGE
PROCESS
DELIVERY
SEND TO USER
END


Leave Process
SAP R/3
Sharepoint
Email Notification
Employee
Approver
System
Fill Leave Application
Make Changes
Screen Message of submission confirmation
Notification of rejection of leave
Notification of approval of leave
Rejected
Approved
Receives an email intimation him regarding the application
Is the leave approval pending for >10 days OR Leave end date < Today
Repeat intimation email after 5 days
yes
No
Approved or rejected
Notification of automatic Approval to Approver & Employee
Yes
Approved
Rejected
Validate with the data
Check for conflicts / return error
Create a locked record in IT2001
Automatic Approval of leave
Unlock the locked record in IT 2001
Leave balance is updated
Mark the locked record in IT2001as Deleted
Rules
When a user opens leave form, Approver is fetched from SAP based on delegation rules.
When there is a need to cancel a leave, the entire leave is deleted. Modifications will be handled as Delete and Reapply.
Deletion of approved Leaves will follow the same approval cycle.
A notification is sent to the approver if a Leave deletion is not approved after 5 days. This will be Automatically approved after 10 days and notification sent to Approver and Employee.

Classical Approaches

Unified Modeling LANGUAGE

Language for:

Visualizing: Graphical models with precise semantics

Specifying: Models are precise, unambiguous and complete to capture all important Analysis, Design, and Implementation decisions.

Constructing: Models can be directly connected to programming languages, allowing forward and reverse engineering

Documenting

Notions for UML Modeling

•Booch …….Grady Booch

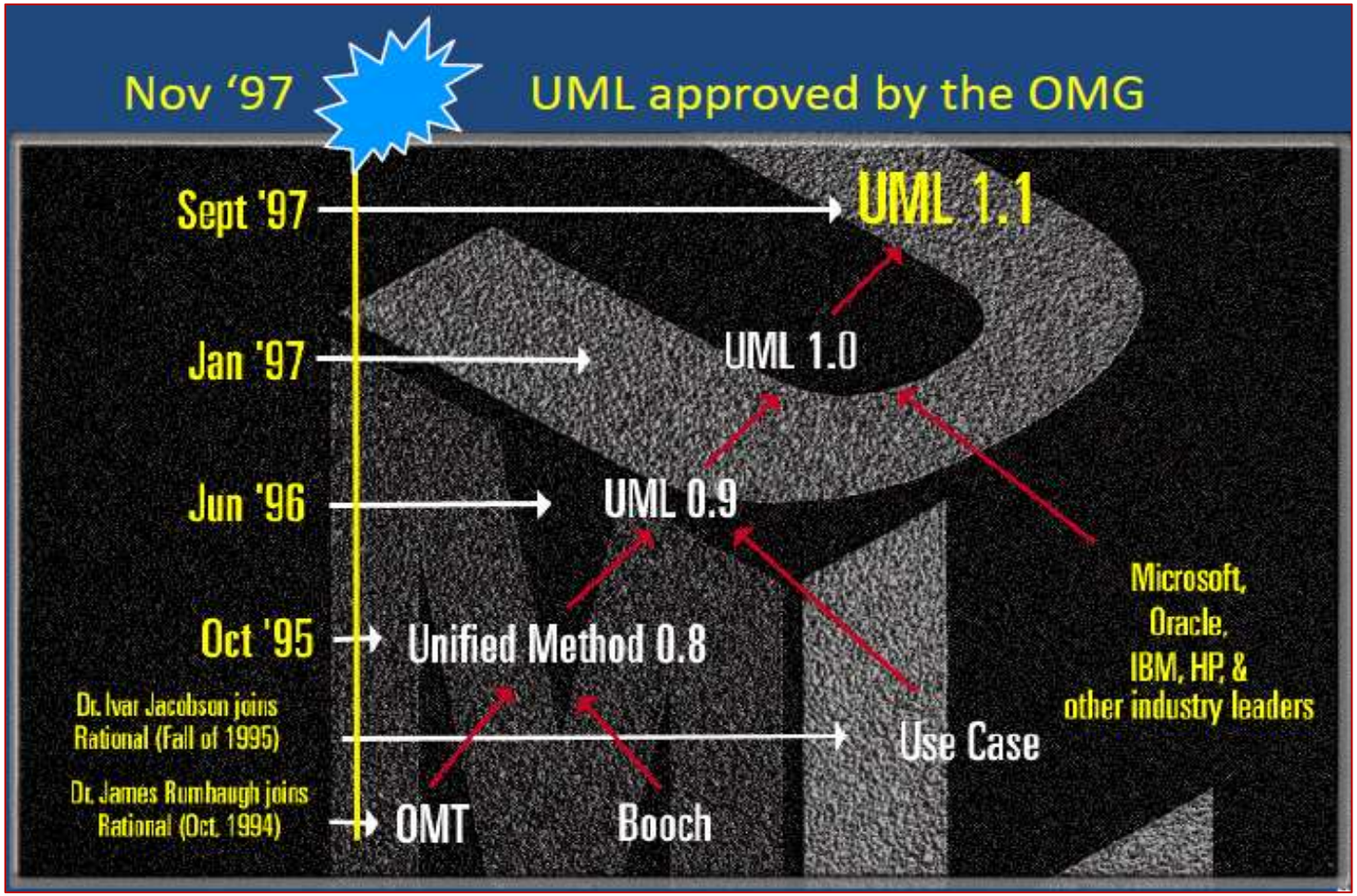
•OMT (Object Modeling Technology). Dr. James Rumbaugh

•Unified Modeling Language (UML)

Booch, OMT, Jacobson

Note: Standards Governing Boards such as ANSI & Object Management Group OMG)

Brief History of UML

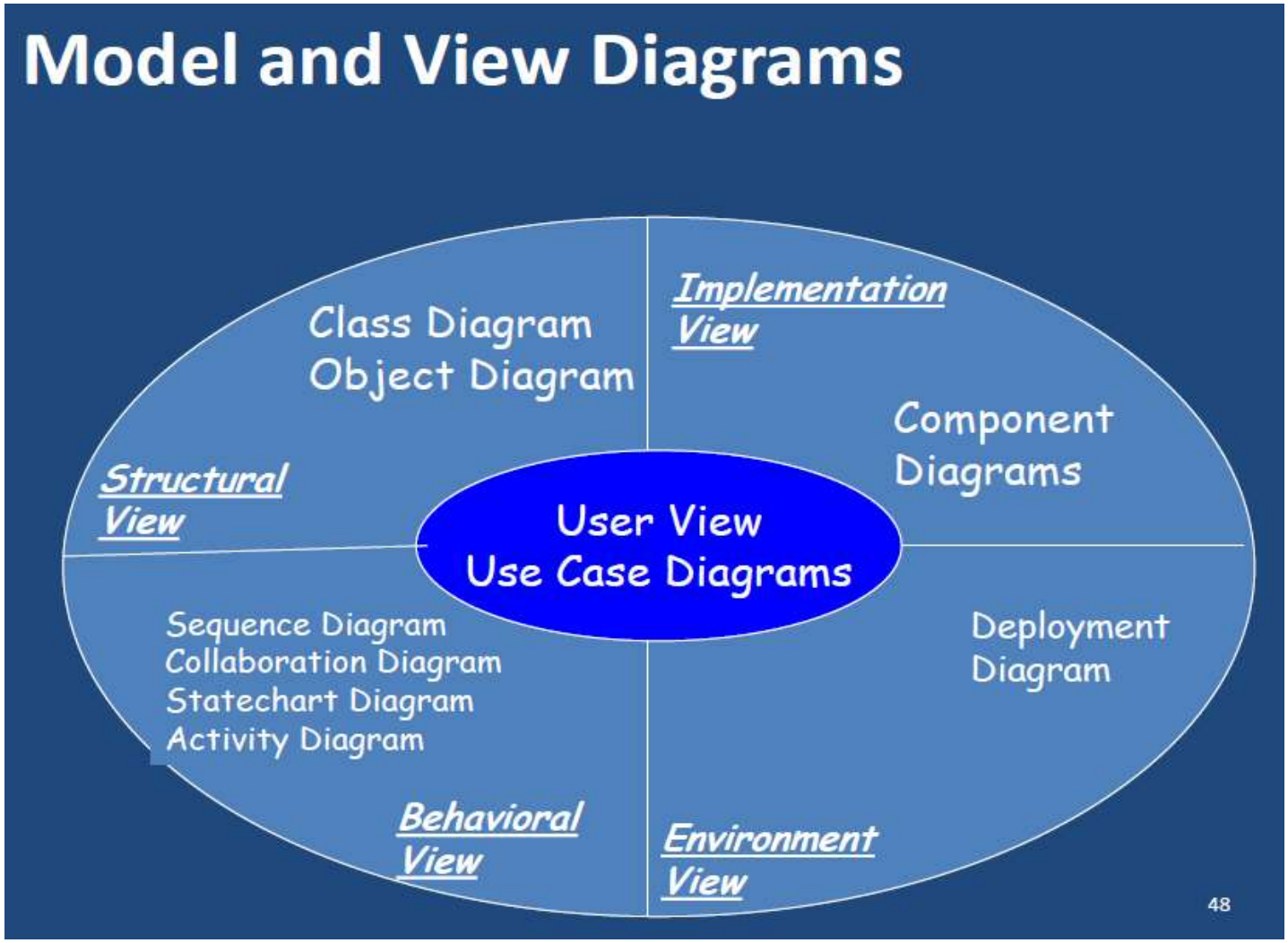


Diagrams in the UML

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram
- Component diagram
- Deployment diagram

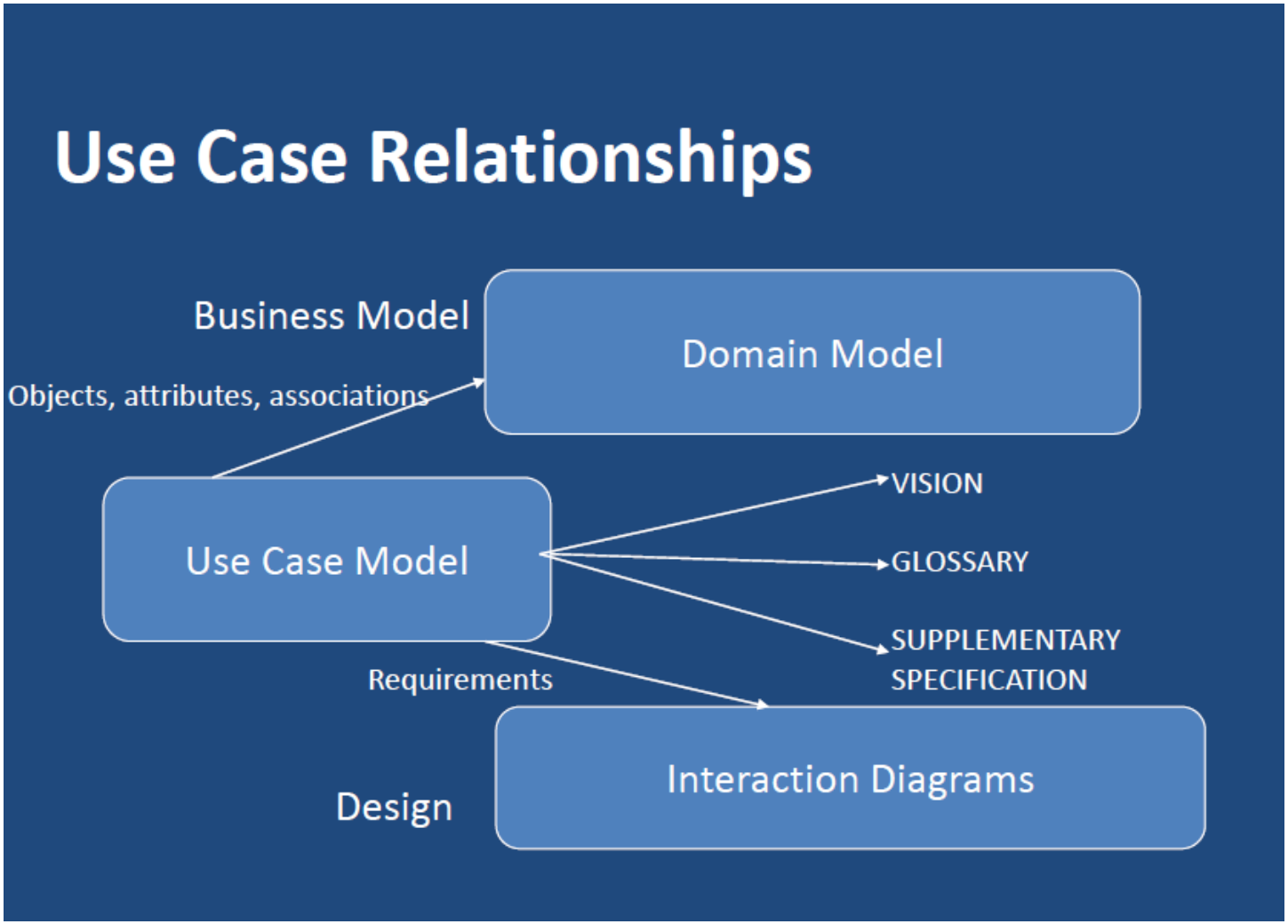
Behavioral Modeling

- Use Cases Diagrams
- Interactions Diagrams
- Activity Diagrams


Model and View Diagrams
Class Diagram
Object Diagram
Implementation View
Component Diagrams
Structural View
User View
Use Case Diagrams
Sequence Diagram
Collaboration Diagram
Statechart Diagram
Activity Diagram
Deployment Diagram
Behavioral View
Environment View
48

PM tutor
Empowering Excellence

USe Cases Diagram


Use Case Relationships
Business Model
Domain Model
Objects, attributes, associations
Use Case Model
VISION
GLOSSARY
SUPPLEMENTARY SPECIFICATION
Requirements
Design
Interaction Diagrams

Benefits of Use Cases

- Captures functional requirements from user's perspective
- Gives a clear and consistent description of what the system should do
- A basis for performing system tests
- Provides the ability to trace functional requirements into actual classes and operations in the system

What is a Use Case?

•Created by Ivar Jacobson (1994)

•"A use case is a sequence of transactions in a system whose task is to yield a measurable value to an individual actor of the system"

•Describes WHAT the system (as a "Black Box") does from a user's perspective

•A set of scenarios tied together by a common user goal

•TheUseCaseModelisNOTaninherentlyobjectoriented modelingtechnique

Use Case Principles

- Describes required functionality in terms of the user –system
- Identifies external actors
- Identifies system boundary
- Describes scenarios of use
- Describes pre/post conditions
- Describes variants/exceptions

Use Case Hints

- Use language of user
- Avoid implementation
- Get good scenarios
- Names single, identifiable and reasonably atomic behavior
- Describes flow of events clearly enough so outsider can follow

Use Case Hints…….

•Factors common behavior by pulling it from other use cases that it includes

•Factors variants by pushes such behavior into other use cases that extend it.

•Show only use cases that are important to understand system

•Show only actors that relate to use cases

•Specify actor names by specific roles

•Use top level diagram to show context

•Decompose top-level use cases to show requirements

•Group common use cases into packages

UML's Use Case Diagrams

•A Use Case model is described in UML (Unified Modeling Language) as one or more Use Case Diagrams (UCDs)

•A UCD has 4 major elements:

–The system described

–The actors that the system interacts with

–The use-cases, or services, that the system knows how to perform

–The relationships between the above elements

Use Case

•A use case is a coherent unit of externally visible functionality provided by a system and expressed by a sequence of messages

•Additional behavior can be shown with parent-child, extend and include use cases

•It is labeled with a name that the user can understand

System

•As part of use-case modeling, the **boundaries of the system** developed must be defined

•A system does not necessarily have to be a software system

•Defining the boundaries of the system is not trivial

–Which tasks are automated and which are manual?

–Which tasks are performed by other systems?

•The entire solution that we supply should be included in the system boundaries

•Incremental releases

•A system in a UCD is represented as a box

•The name of the system appears above or inside the box

Actor

•Someone or something that interacts with the system (exchanges information with the system)

•An actor represents a role played with respect to the system, not an individual user of the system

Example:

–Policeman –Enters data

–Supervisor –Allowed to modify/erase data

–Manager –Allowed to view statistics.

•A single user may play more than one role

•Actors have **goals**:

–Add a Traffic Violation

–Lookup a Traffic Violation

•Actors don't need to be human

–May be an external system that interfaces with the developed system

•An actor has a name that reflects its role

Types of Actors

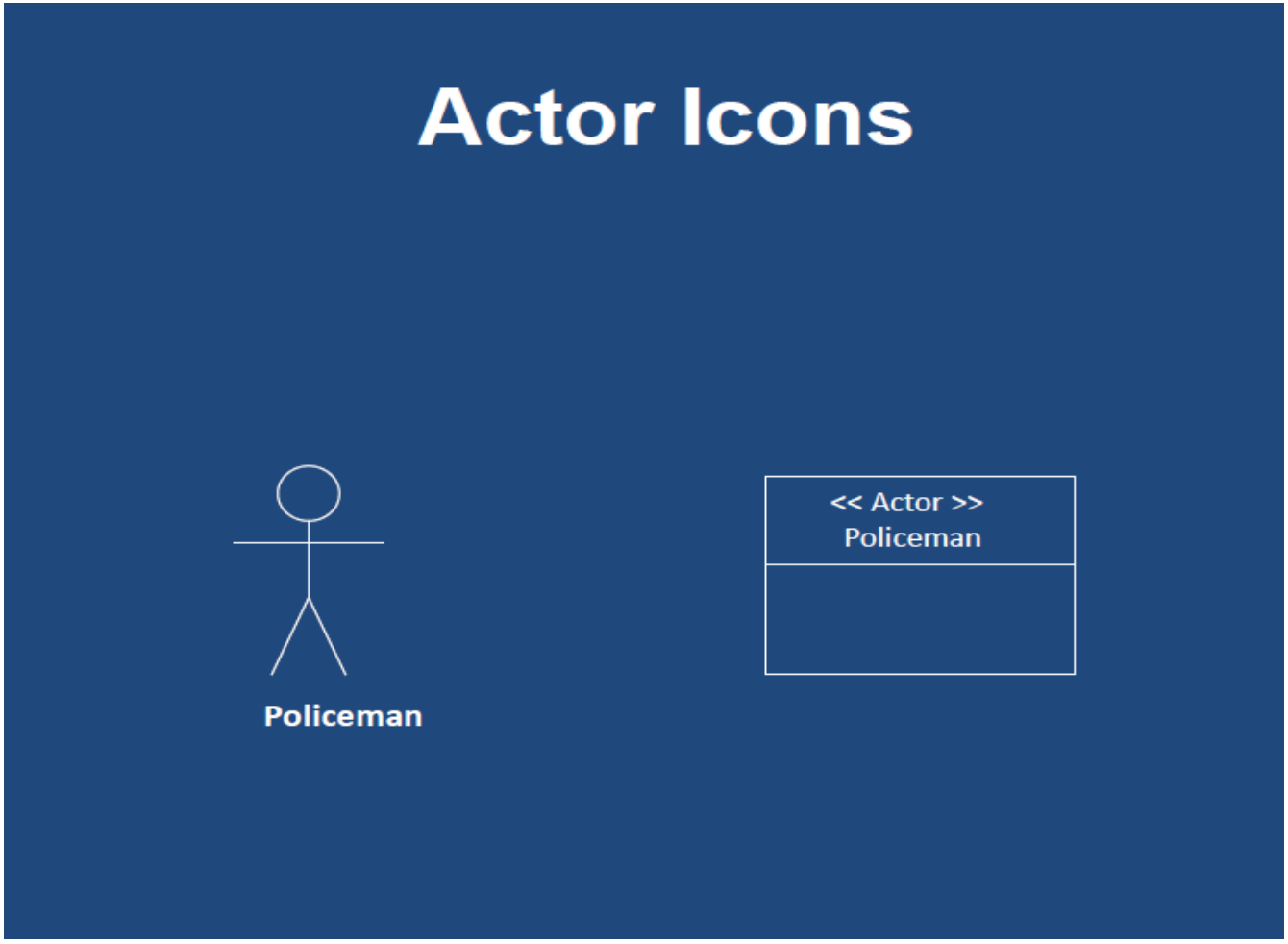
- Primary Actor
  - Has goals to be fulfilled by system

- Supporting Actor
  - Provides service to the system

- Offstage Actor
  - Interested in the behavior, but no contribution

- In diagrams, Primary actors go on the left and others on the right.

Types of Actors



Relationships between Actors

•When several actors as part of their roles, also play a more generalized role, it is described as **generalization**

•The behavior of the general role is described in an actor super-class

•Thespecializedactorsinheritthebehaviorofthesuper-classandextenditinsomeway

•Relationships between actors are not always necessary

Golden Rule of Use-Case Names

•Each use case should have a name that indicates what value (or goal) is achieved by the actor's interaction with the system

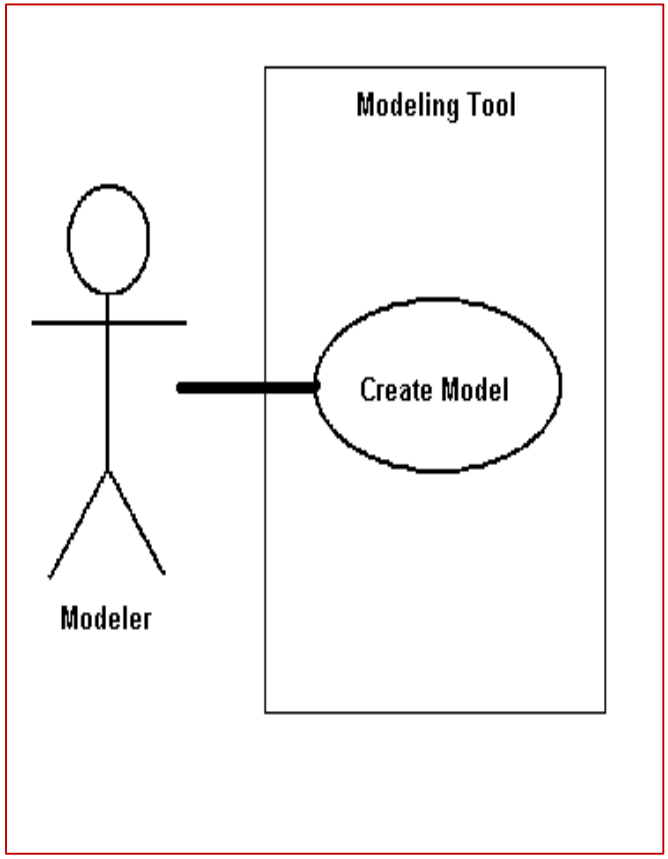
•Here are some good questions to help you adhere to this rule:

Why would the actor initiate this interaction with the system?
What goal does the actor have in mind when undertaking these actions?
What value is achieved and for which actor?

Use-Case Name Example

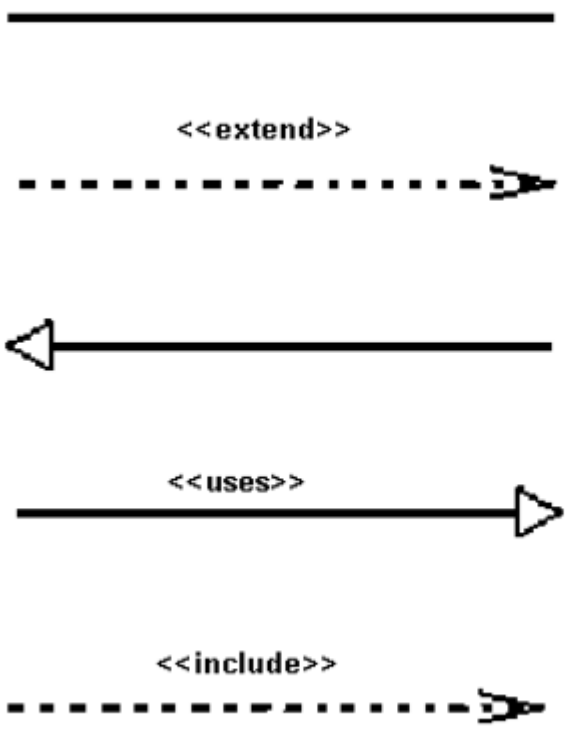
- **Excellent**–Purchase Concert Ticket
- **Very Good**–Purchase Concert Tickets
- Good–Purchase Ticket (insufficient detail)
- **Fair**–Ticket Purchase (passive)
- **Poor**–Ticket Order (system view, not user)
- **Unacceptable**–Pay for Ticket (procedure, not process)

Association Relationship



- An association is the communication path between an actor and the use case that it participates in
- It is shown as a solid line
- It does not have an arrow, and is normally read from left to right
- Here, the association is between a Modeler and the Create Model use case

Relationships in Use Cases



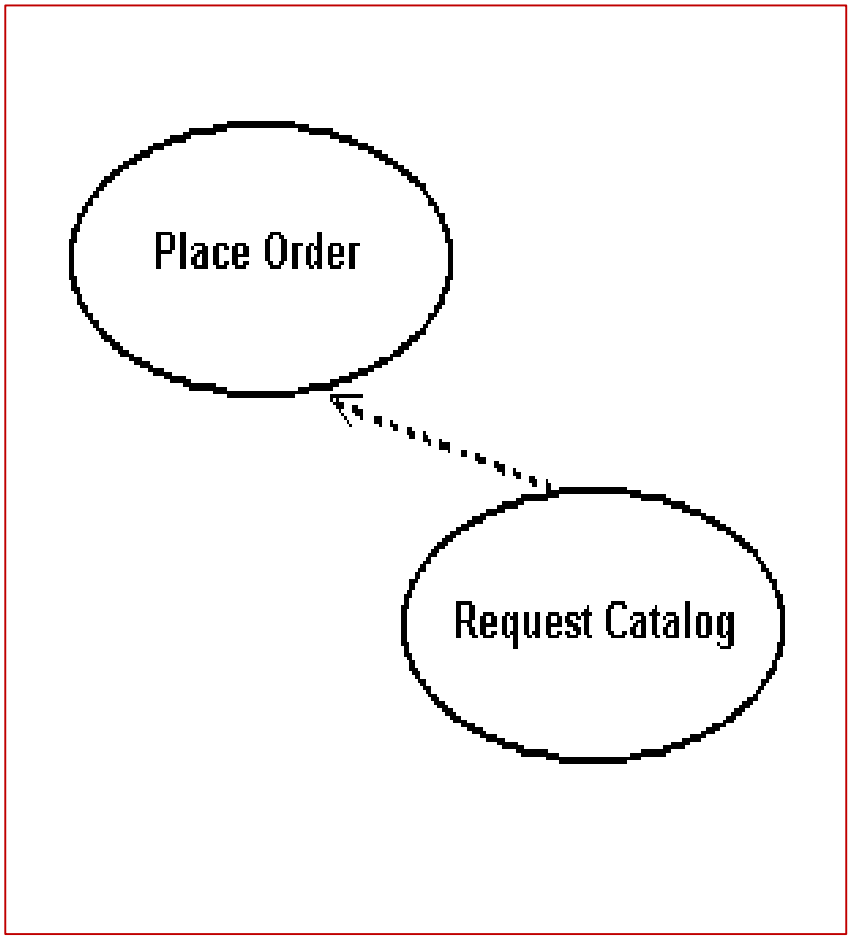
- There are several Use Case relationships:
- Association
- Extend
- Generalization
- Include

Extend Relationship

•Extend puts additional behavior in a use case that does not know about it.

•It is shown as a dotted line with an arrow point and labeled<<extend>>

•In this case, a customer can request a catalog when placing an order

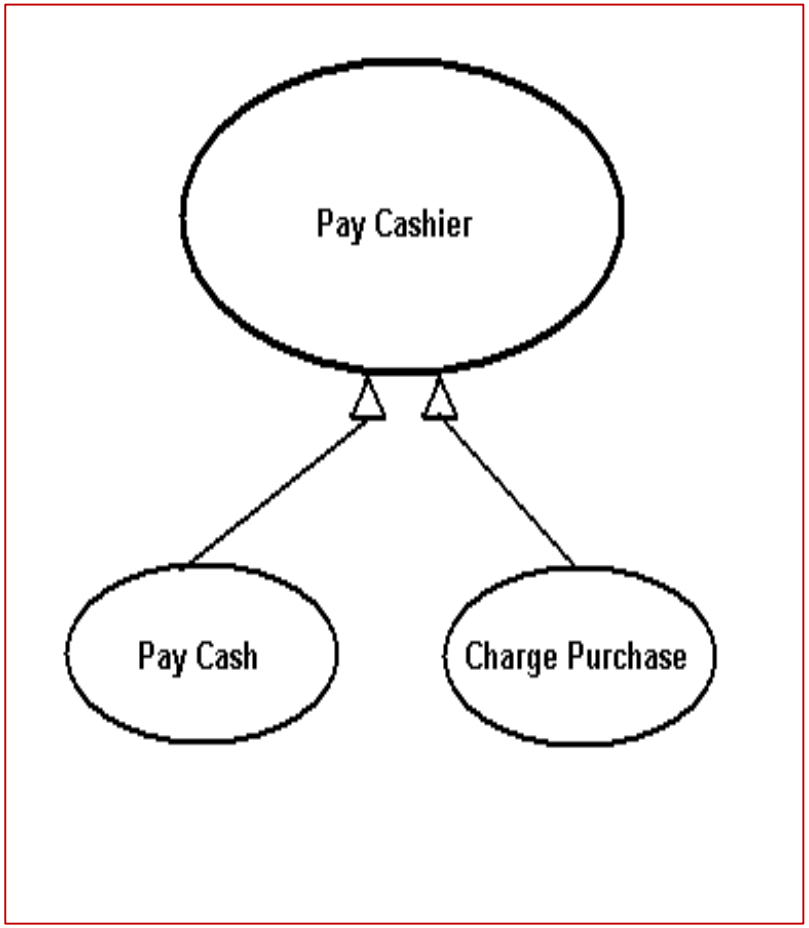


Use Case Generalization

•Generalization is a relationship between a general use case and a more specific use case that inherits and extends features to it

•It is shown as a solid line with a closed arrow point

•Here, the payment process is modified for cash and charge cards

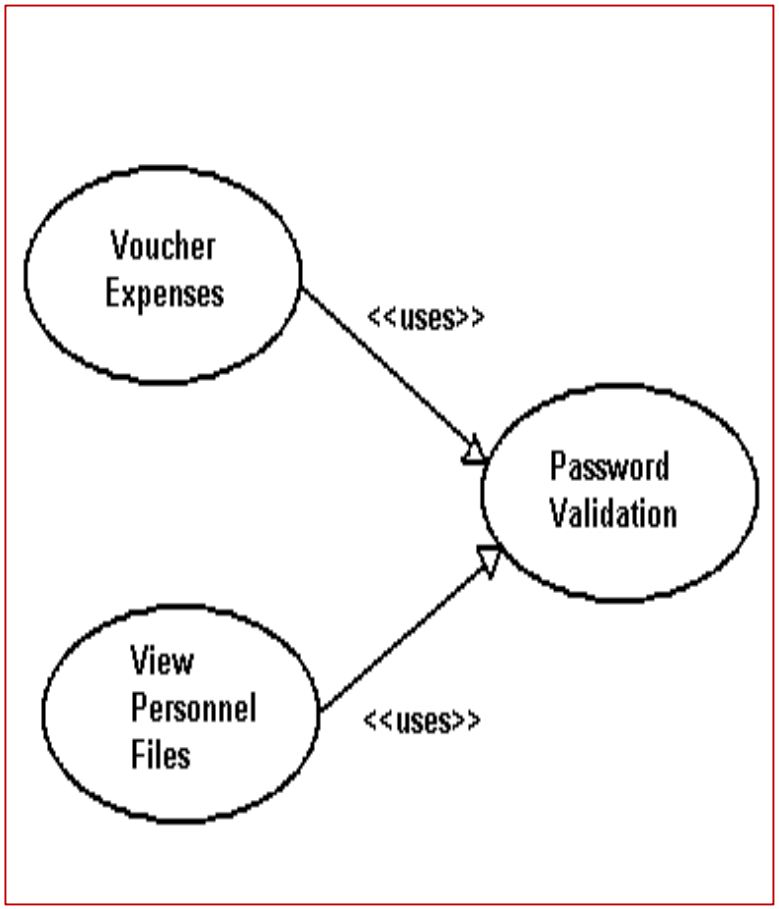


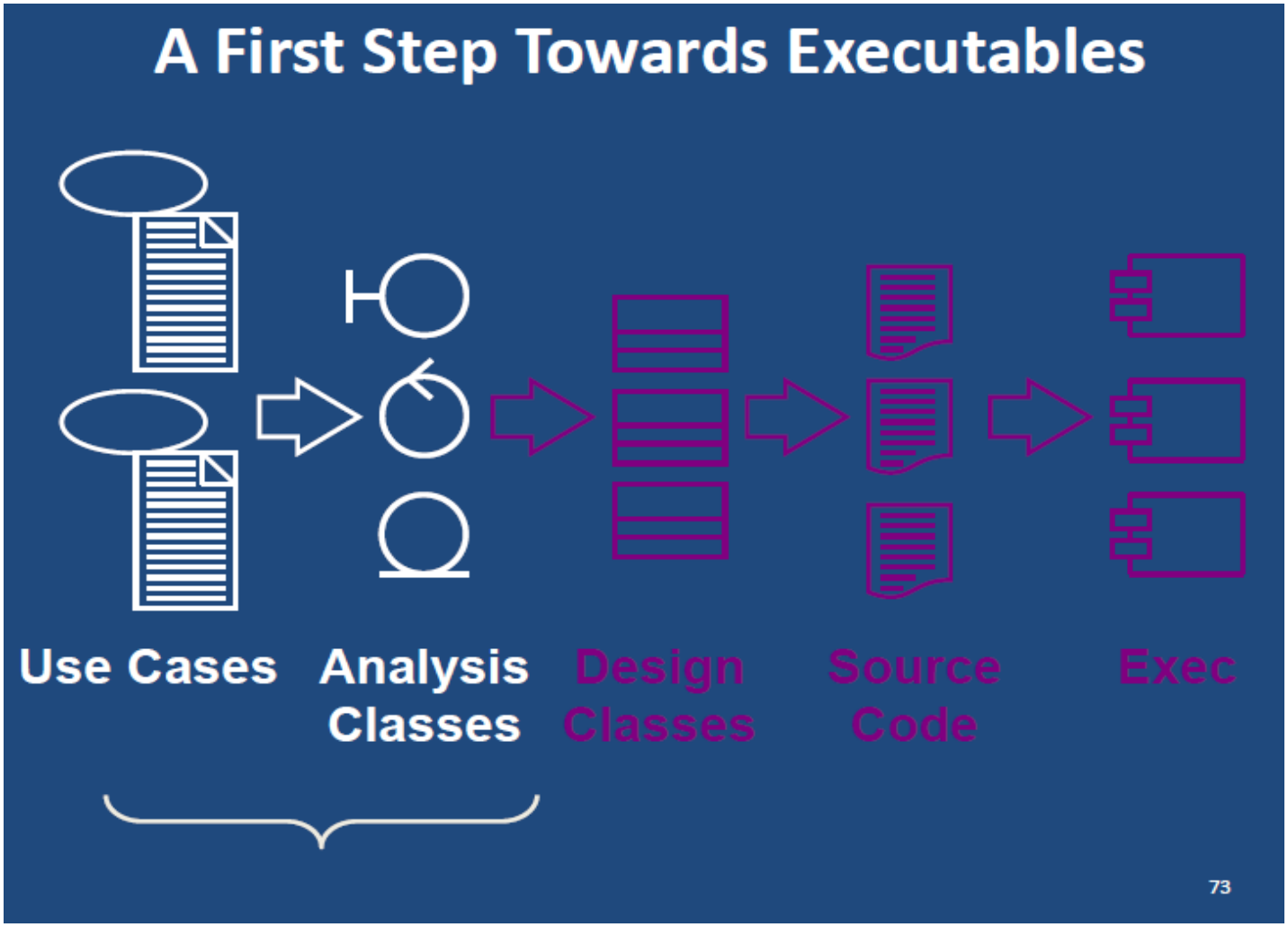
Include or Uses Relationship

•When a use case uses another process, the relationship can be shown with the uses relationship

•This is shown as a solid line with a closed arrow point and the <<uses>> <<includes>>keyword

•Here different system processes can use the logon use case




A First Step Towards Executables
Use Cases
Analysis Classes
Design Classes
Source Code
Exec
73