

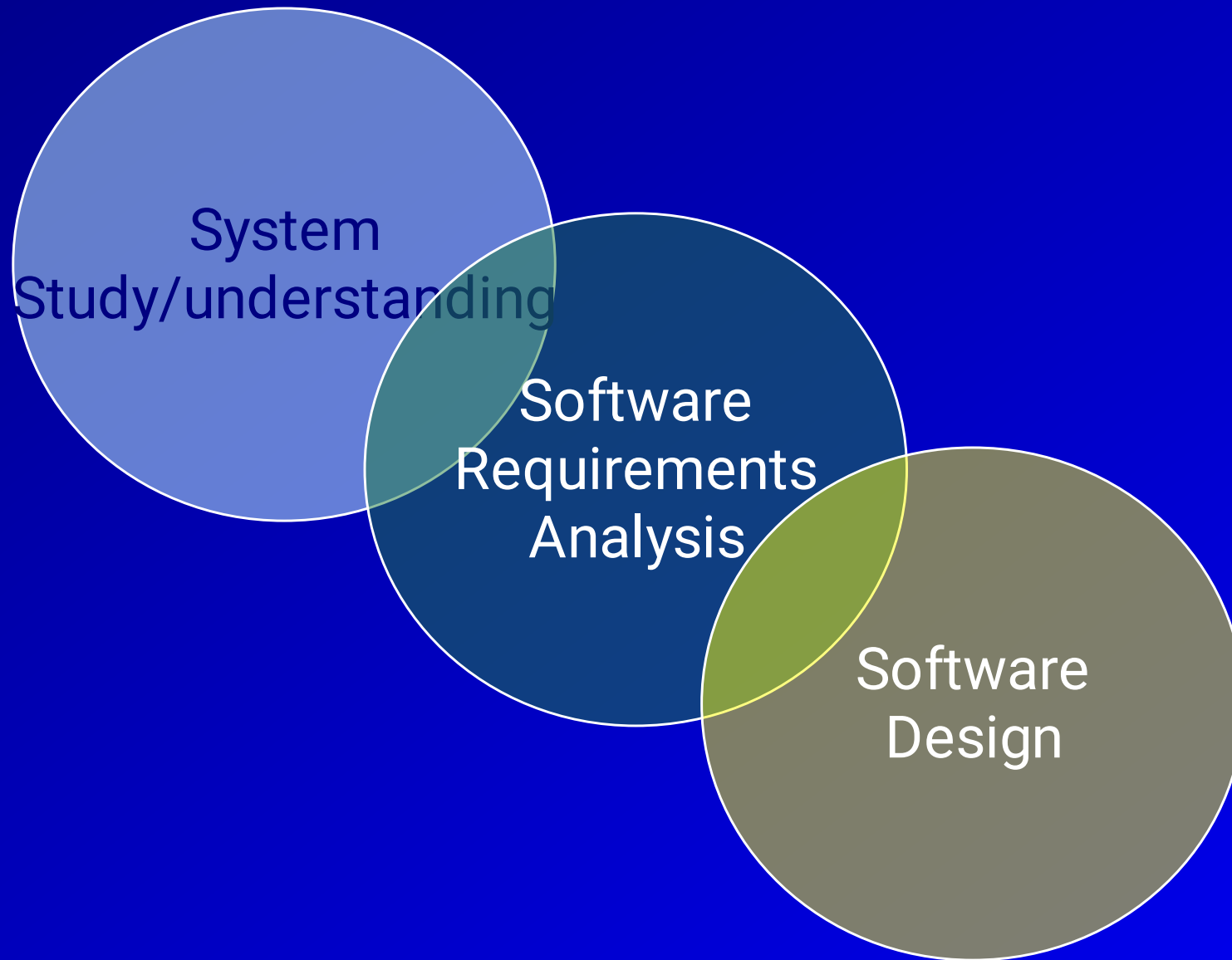
Requirements Analysis

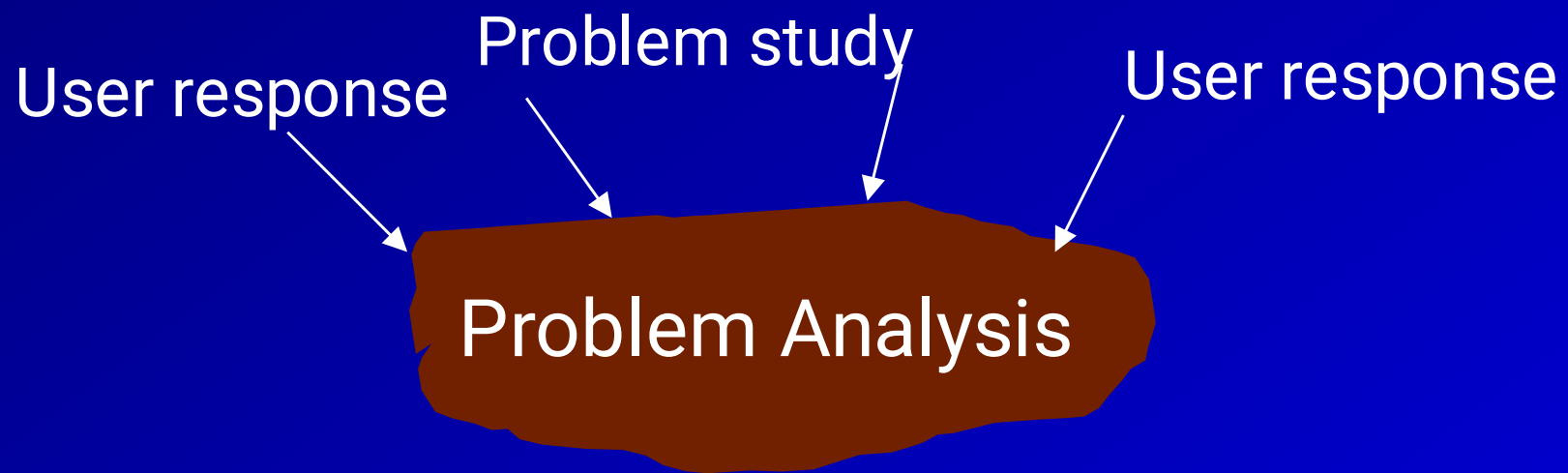
Concepts & Principles

Requirements Analysis and Specification

F Many projects fail:

- because they start implementing the system:
- without determining whether they are building what the customer really wants.





A relatively good understanding of user's requirements

System Description

Software Requirements Specification

SW Requirements Analysis

- F A study of **user needs**, for a problem to arrive at a **definition of WHAT the software will do** without describing how it will do it.
- F A **Software Requirements Specification (SRS)** is a document containing software requirements specification for a system.

Requirements Analysis Activities

- F Study System Specification, SW Project plan & feasibility report
- F Problem Recognition
- F Evaluation & understanding
- F Modeling
- F Specification
- F Review

Requirement Elicitation

Fact Gathering Techniques

- Interviews
- Questionnaires
- Analyzing documents
- Observation
- study...

Requirement Elicitation contd.

Facilitated Application Specification Technique

- F Meeting at neutral site
- F Rules for preparation & participation
- F Agenda – formal + informal
- F Facilitator controls meeting
- F Definition mechanism
- F Identify problem, propose solution, negotiate approaches, specify set of solution requirements

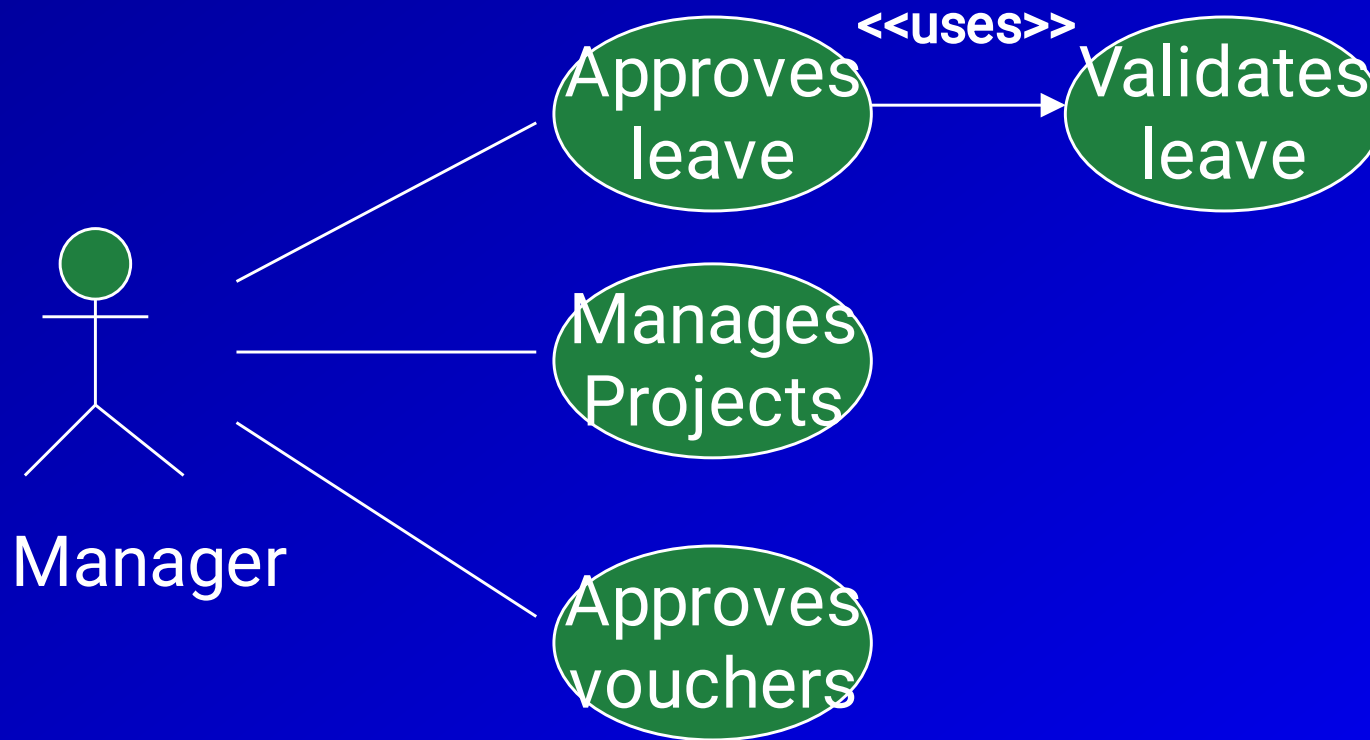
Requirement Elicitation contd.

Quality Function Deployment - quality management technique

- F Normal requirements
 - F Expected requirements
 - F Other requirements
-
- Function Deployment : function values
 - Information Deployment : data objects + events
 - Task deployment : system behavior
 - Value Analysis : requirement priorities

Requirement Elicitation contd.

Use Case diagram (usage scenario of HR department)



Requirement Elicitation contd.

- F Scenarios viewed differently by different actors
- F Use U_C Modeling : Priority value assigned for each use case
- F System functionality with priorities

Analysis Principles

- F Information domain must be represented thoroughly
- F Functions to perform, must be defined and modeled properly
- F Behavior, as consequence of external events must be represented correctly
- F Models depicting information/data, function & behavior must be partitioned to uncover detail in layered fashion (i.e. step by step)
- F Analysis process should move from essential information (i.e. most abstract level) to implementation (most detailed) level.

Analysis Principles

- F Understand the problem clearly before creating analysis and design model
- F Develop prototypes
- F Record the origin & the reason for every requirement (trace)
- F Use multiple views of requirements
- F Rank (prioritize) all requirements
- F Work to eliminate ambiguity in requirements if any

Analysis Principles (contd.)

Information Domain

- F Information contents & relationships (Data Model)
- F Information Flow (data flow)
- F Information structure (data dictionary) and internal organization of data (data structure)

Analysis Principles (contd.)

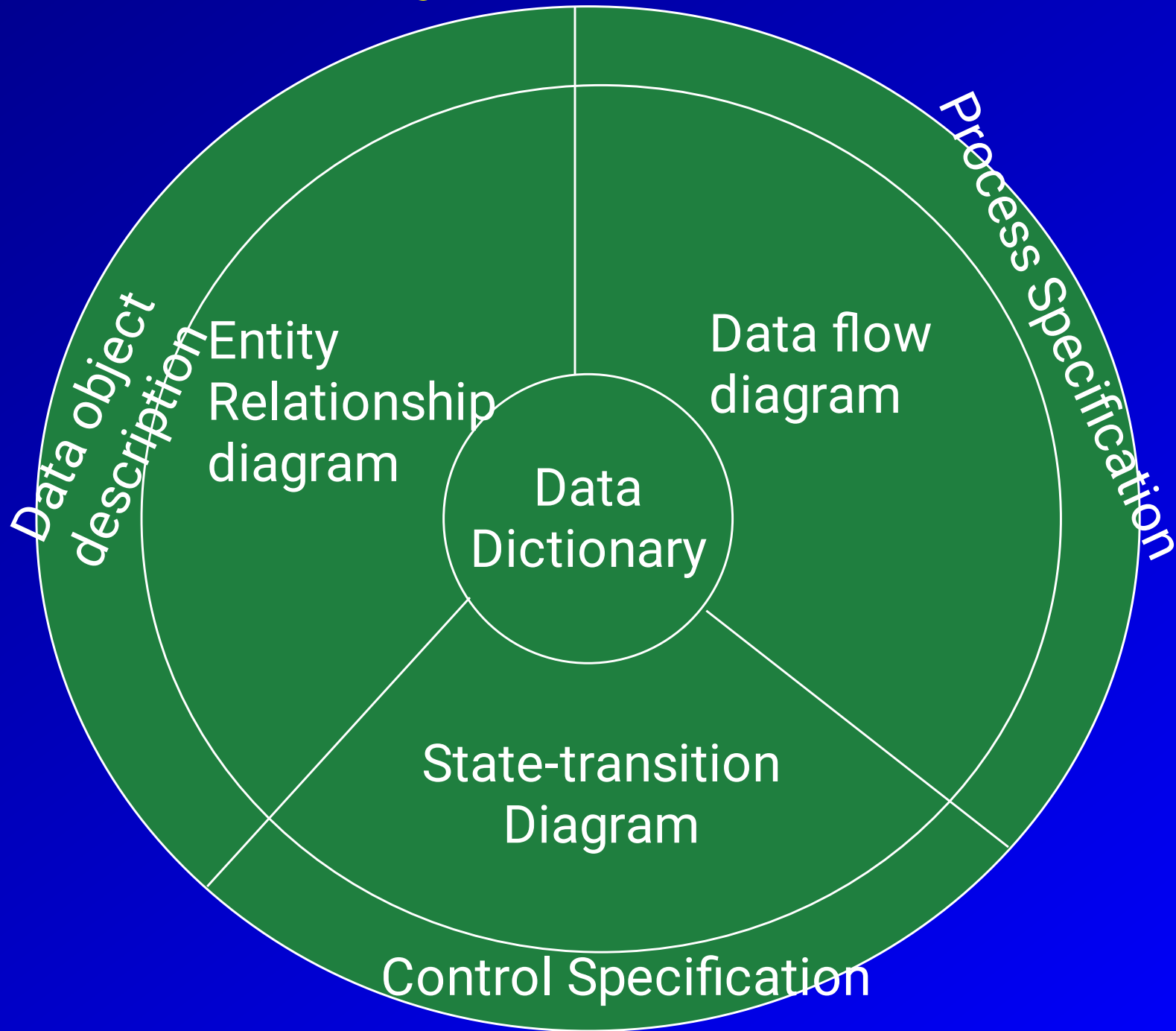
Various models of Structured Analysis

- F Functional Model (DFD)
- F Structural Model (ERD)
- F Behavioral Model (time dependant behavior) (STD)

Modeling

- F A model is an abstract representation of a view of the system
- F Build different models of the system to view it from various ways/angles
- F Benefits of modeling
 - focuses on important features of the system while avoiding less important features
 - discuss changes and corrections to the user's requirements with low cost and minimal risk
 - verify that Analyst correctly understands the user's requirements

Analysis Models



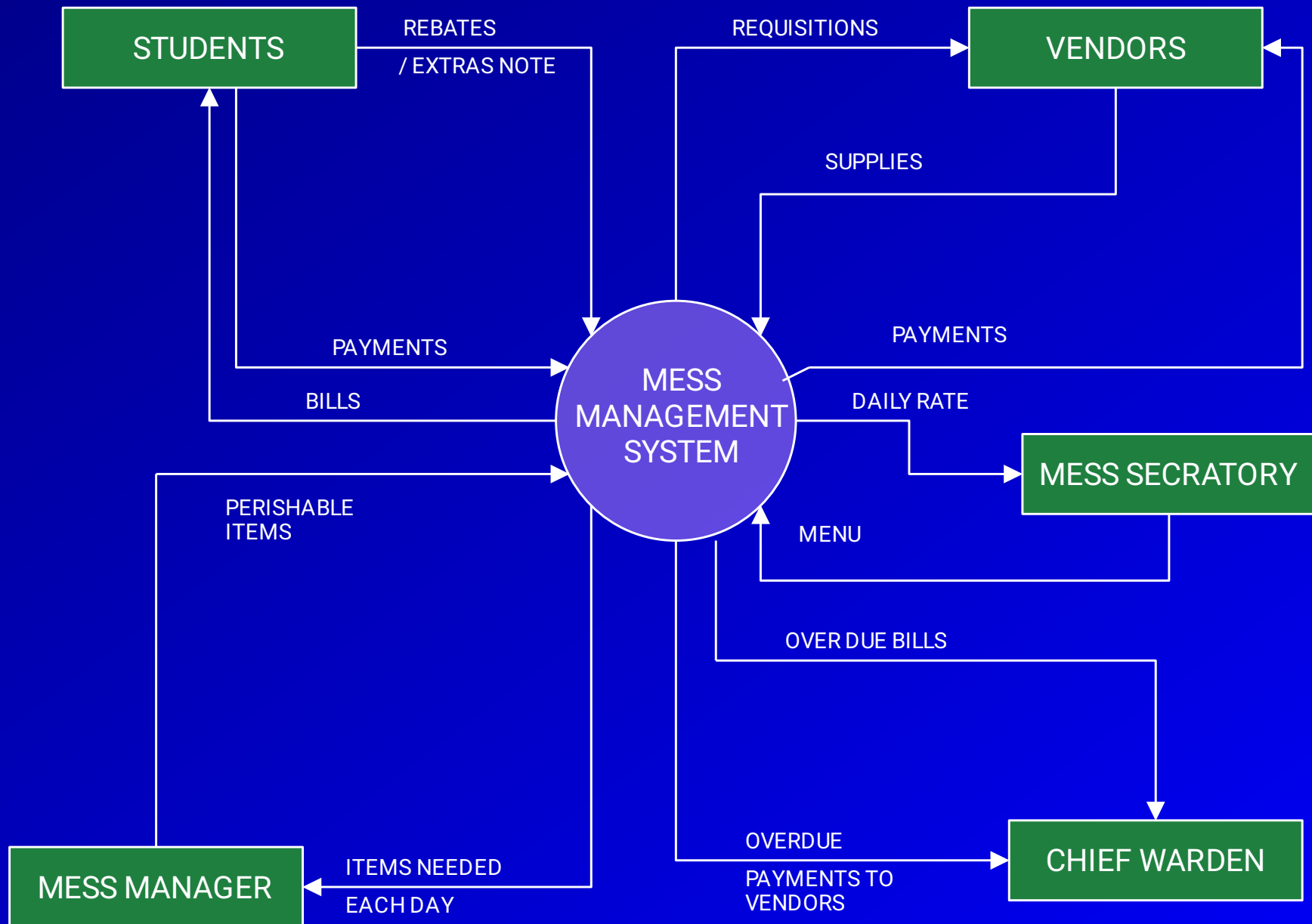
Analysis Models

- F Data Flow Diagram functional view
- F Entity Relationship Stored Data view
Diagram
- F State Transition Time-dependent
Diagram behavioral view

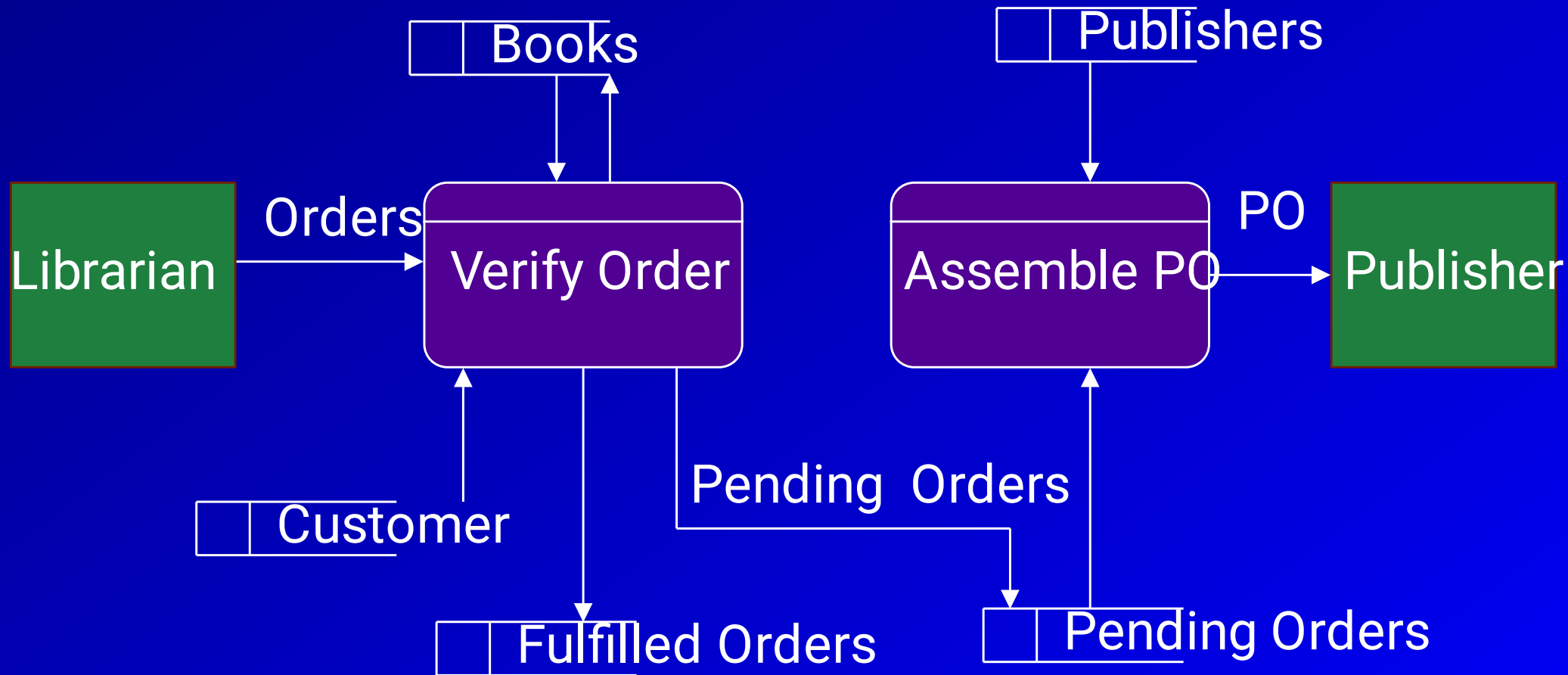
Data Flow Diagram (DFD)

- F A DFD depicts the flow of information within a system and between the system and the outside world.
- F It does not show sequence and control information.
- F A Data Flow Diagram (DFD) contains:
 - External entities, Processes, Data flows, Data stores

LEVEL 0 DFD for Mess Management System



DFD Example (LIS Level 2)



DFD Primitives









External Entity represents a source or sink for data. These are usually outside the scope of the area under study.

Process transforms or manipulates data

Data flow carries data between an entity and a process, a process and a store, or between two processes

Data Store represents a repository of data

Connector symbols are used if the diagram spans multiple pages.

	Yourdon and Coad	Gane and Sarson
External Entity		
Process		
Data Store		
Data Flow		

Types of DFDs

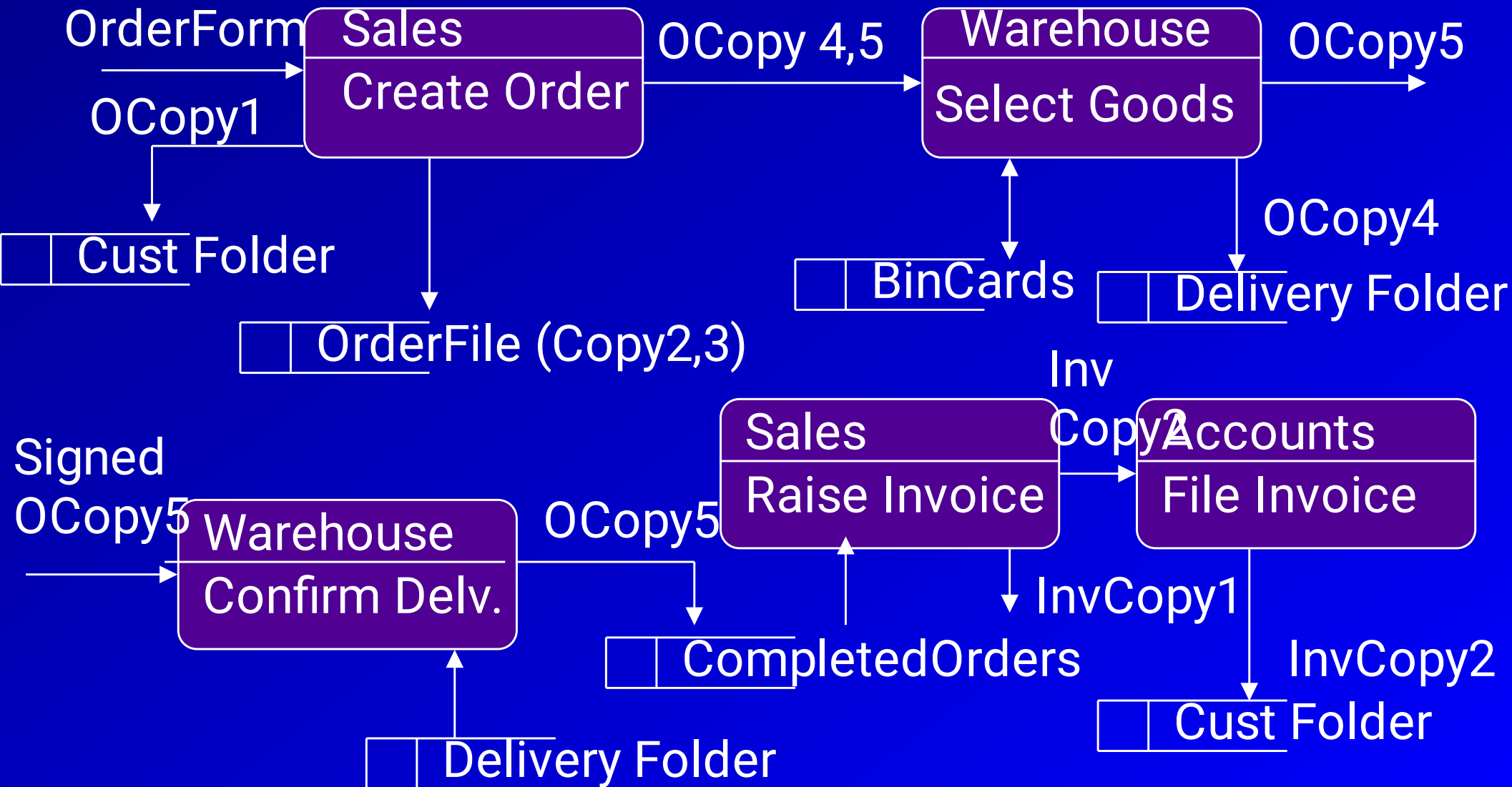
Current Physical DFD

produced in the early stages of analysis
to describe the existing system.

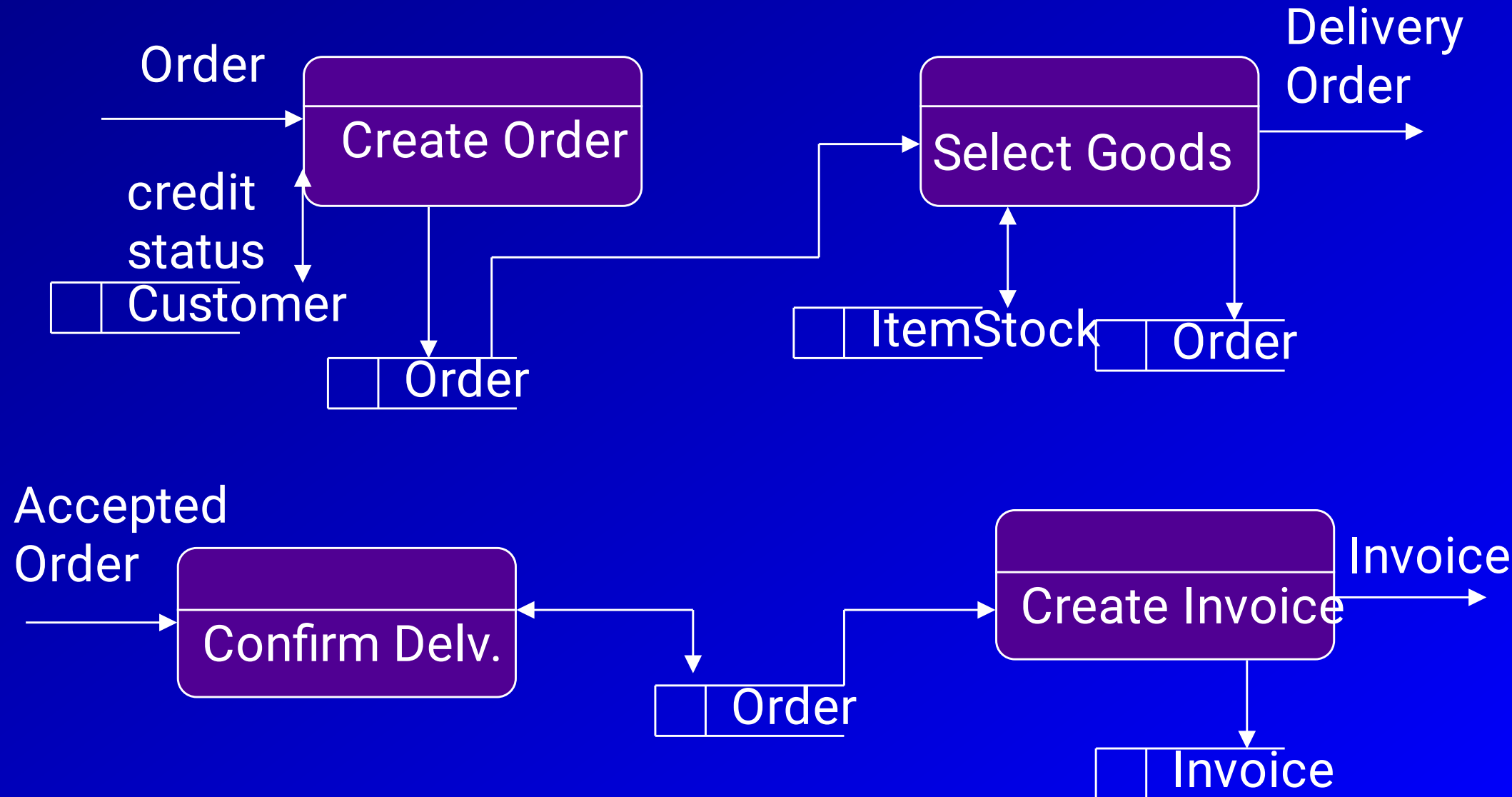
Current Logical DFD

derived from the above, after removing physical elements.

Example: Current Physical DFD



Example: Current Logical DFD (online shopping store Level 2)

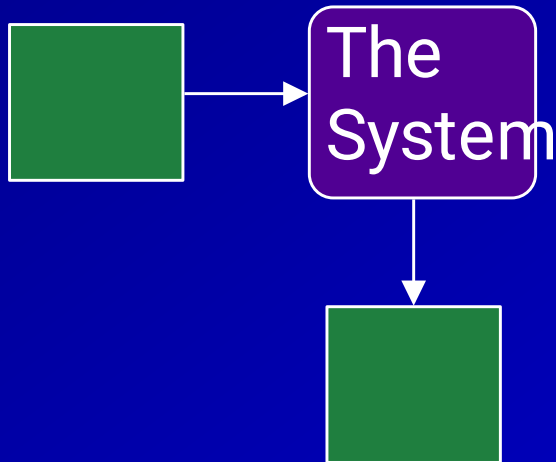


DFD Hierarchy

- F DFD can be used at different levels of abstraction.
- F The highest level DFD is called a **Context Diagram**
- F A process may be exploded into a lower level DFD to show more detail.
- F In such a case, the net input and output flows must be balanced across the DFD levels.
- F When decomposing a DFD, you must protect inputs to and outputs from a process at the next level of decomposition. This is called **balancing**.
- F The lower level DFD may have data stores that are local to it, and not shown in the higher level DFD.

Context Diagram (DFD Level 0)

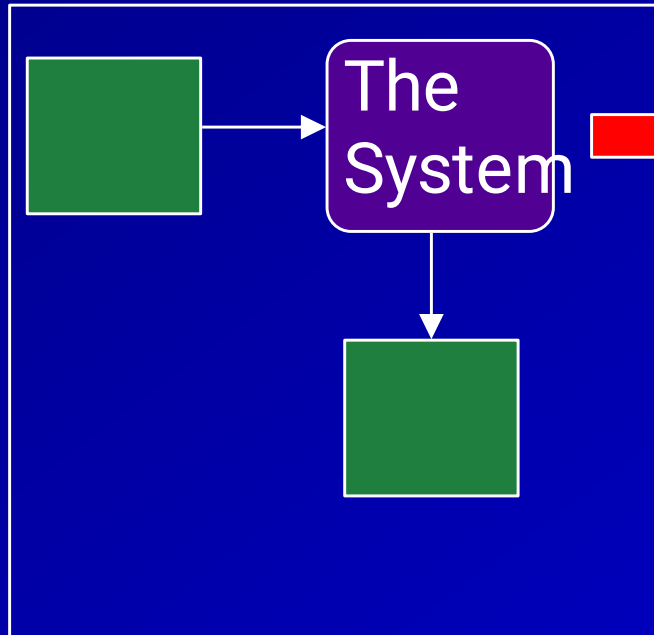
Example: Context Diagram



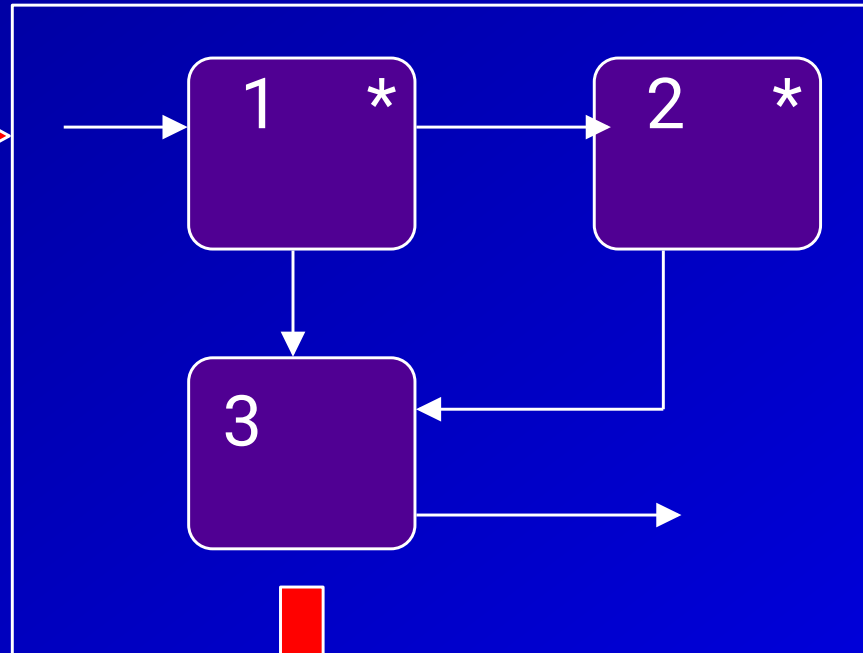
- Represents an entire system as a single process
- Shows the scope of the system
- Highlights the interfaces between the system and the outside terminators (external entities)

Leveled DFDs

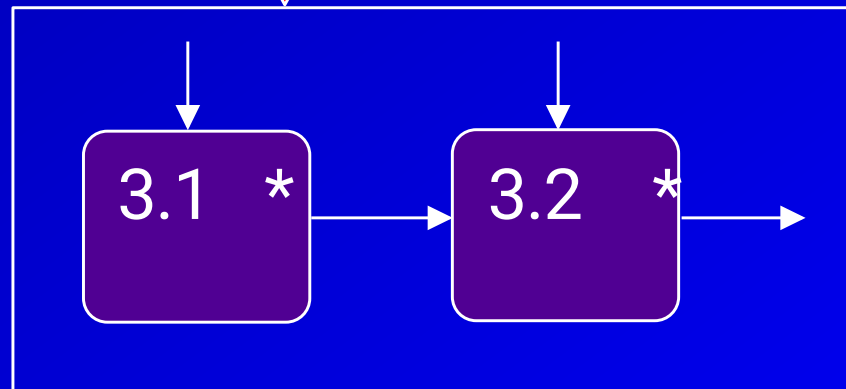
Context Diagram



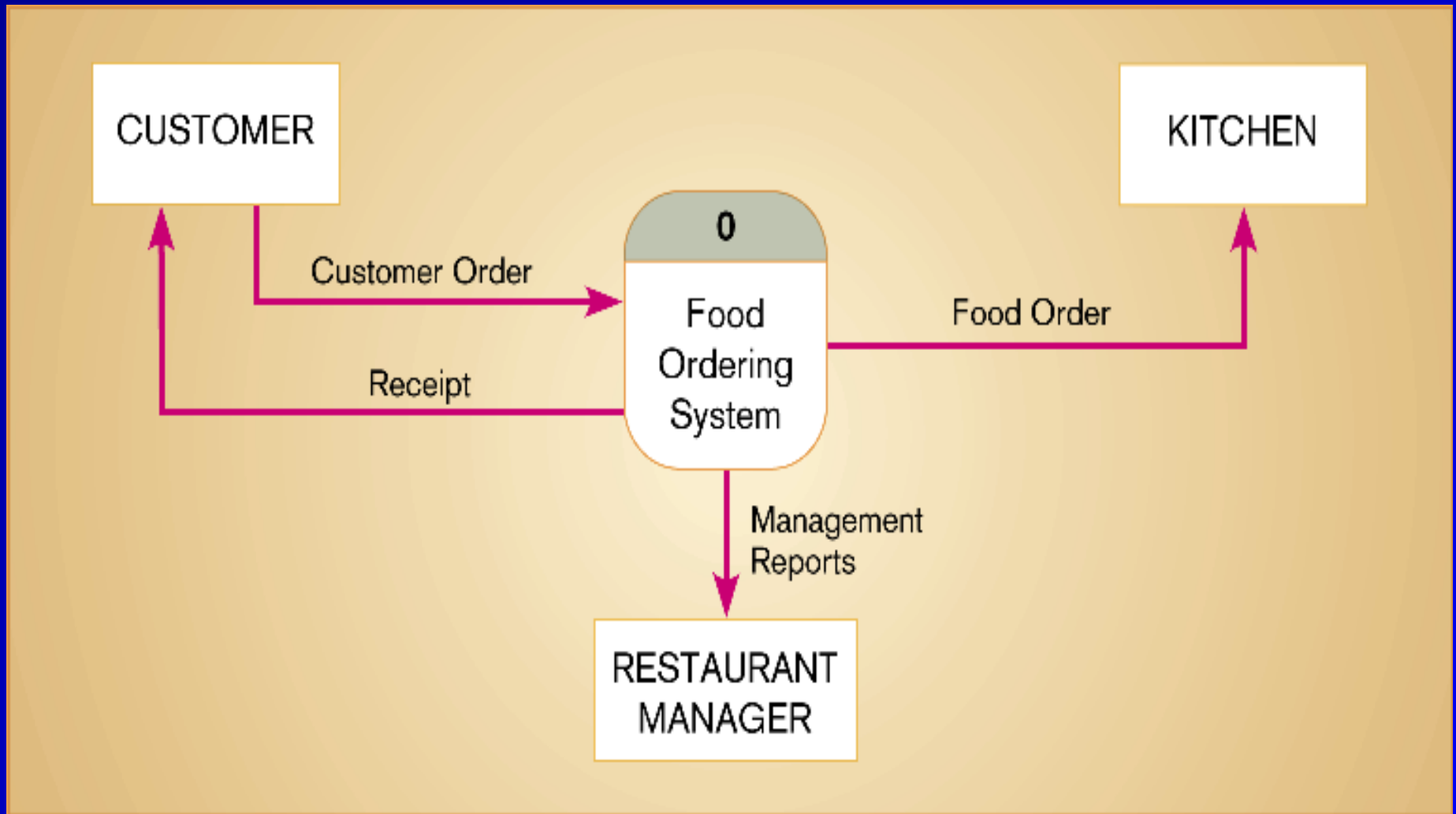
Level-1 DFD



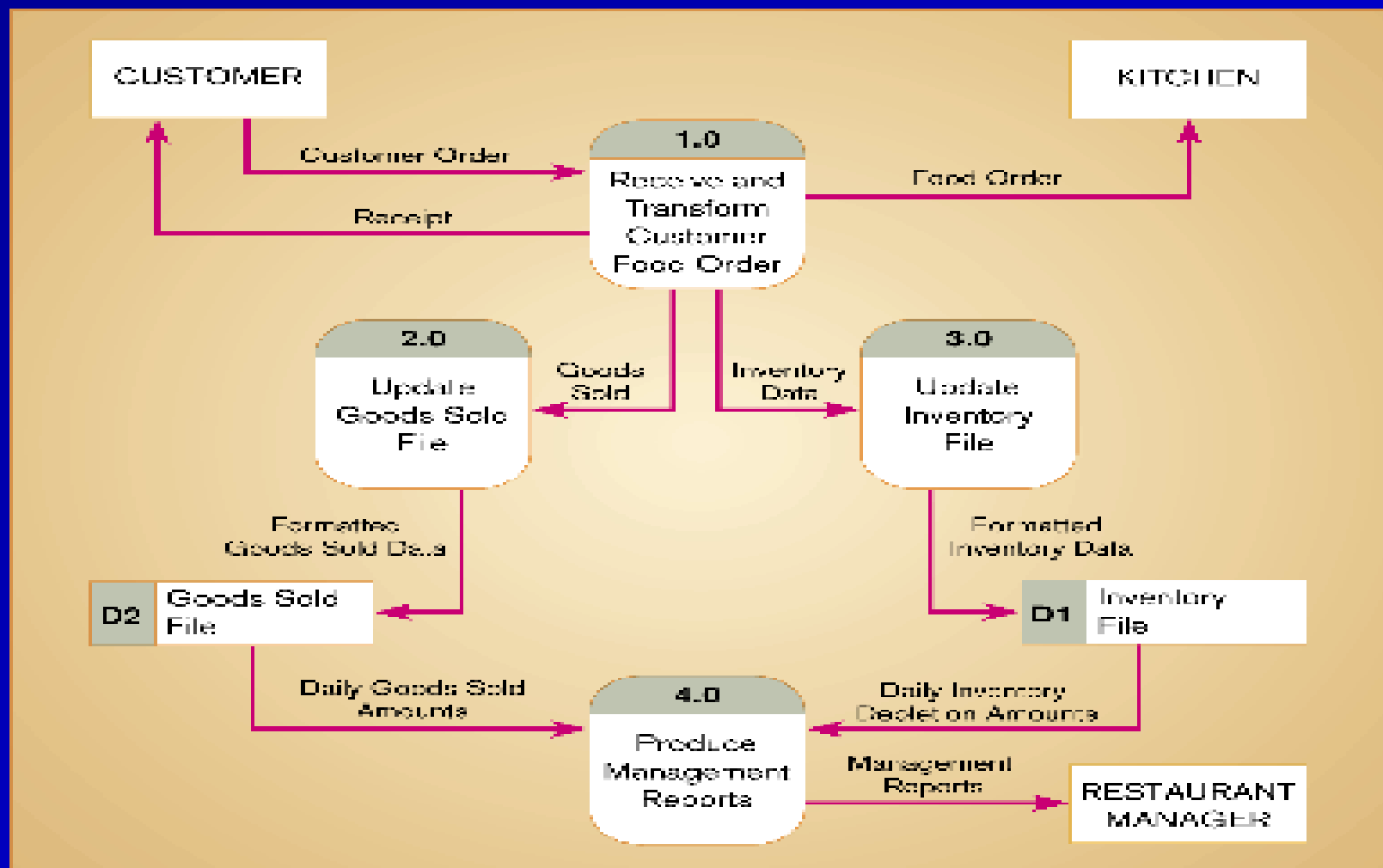
Level-2 DFD



Context diagram of food ordering system



Level-1 DFD of food ordering system



DFD Guidelines

- F Ideally, a DFD at any level, should have no more than half of a dozen processes and related stores.
- F Choose meaningful names for processes, flows, stores, and terminators
 - use active unambiguous verbs for process names such as Issue, Purchase, Create, Calculate, Compute, Produce, etc.
- F **Number** the processes

DFD Guidelines (continue..)

- F Make sure that DFD is internally consistent
 - avoid infinite sinks
 - avoid spontaneous generation of processes
 - beware of unlabeled flows and unlabeled processes
 - beware of read-only or write-only stores

DFD Guidelines (continue..)

- F Avoid overly complex DFDs
- F Balance DFDs across the levels
- F Data flows and stores must be described in the **Data dictionary**.
- F Processes must be supported either by a lower level DFD or a **mini-specification** (algorithm).

Lowest Level Processes in a DFD

- F A process that cannot be exploded further has an associated **mini specification**.
- F Mini spec can be in:
 - Structured English
 - Decision Table/Decision Tree

Example: Process Specification in Structured English (with proper indentation)

```
FOR EACH item in the Indent DO
  IF Item is on Rate-Contract
    prepare PO to Rate-Contract vendor
  ELSE IF Item is proprietary
    prepare PO to manufacturer
  ELSE
    prepare regular PO to registered vendor
END FOR
```

Modeling Logic with Decision Tables

- F A matrix (tabular) representation of the logic of a **decision**
- F Specifies the possible **conditions** and the resulting **actions**
- F Best used for complicated decision logic

Modeling Logic with Decision Tables

F Consists of three parts

- Condition stubs

- u Lists condition relevant to decision

- Action stubs

- u Actions that result from a given set of conditions

- Rules

- u Specify which actions are to be followed for a given set of conditions

Modeling Logic with Decision Tables

- F Standard procedure for creating decision tables
 - Name the **condition** and values each condition can assume
 - Name all possible **actions** that can occur
 - List all **rules**
 - Define the actions for each **rule**
 - Mark each **entry in the table**

F

A decision table is a table composed of rows and columns, separated into **four separate quadrants**.

F

Conditions

Condition Rules

F

Actions

Action Entries

F The upper left quadrant contains the **conditions**.

F The upper right quadrant contains the **condition rules** or alternatives. The lower left quadrant contains the **actions** to be taken and the lower right quadrant contains the **action rules entries**.

Example decision table for **payroll system** example
 S- regular **S**erving employee, H- **H**ired employee on contract

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

Example: Decision Table (customer care system)

Conditions								
Good payment record	y	y	y	y	n	n	n	n
Order Cost > Rs 1 lac	y	y	n	n	y	y	n	n
Membership > 5yrs	y	n	y	n	y	n	y	n
Actions								
Priority treatment	X	X	X		X			
Normal treatment				X		X	X	X

Process Description Tools

F Structured English

- Might look familiar because it resembles pseudocode

SAMPLE OF A SALES PROMOTION POLICY:

- Preferred customers who order more than \$1,000 are entitled to a 5% discount, and an additional 5% discount if they used our charge card.
- Preferred customers who do not order more than \$1,000 receive a \$25 bonus coupon.
- All other customers receive a \$5 bonus coupon.

STRUCTURED ENGLISH VERSION OF THE SALES PROMOTION POLICY:

```
IF customer is a preferred customer, and
    IF customer orders more than $1,000 then
        Apply a 5% discount, and
        IF customer uses our charge card, then
            Apply an additional 5% discount
    ELSE
        Award a $25 bonus coupon
ELSE
    Award a $5 bonus coupon
```

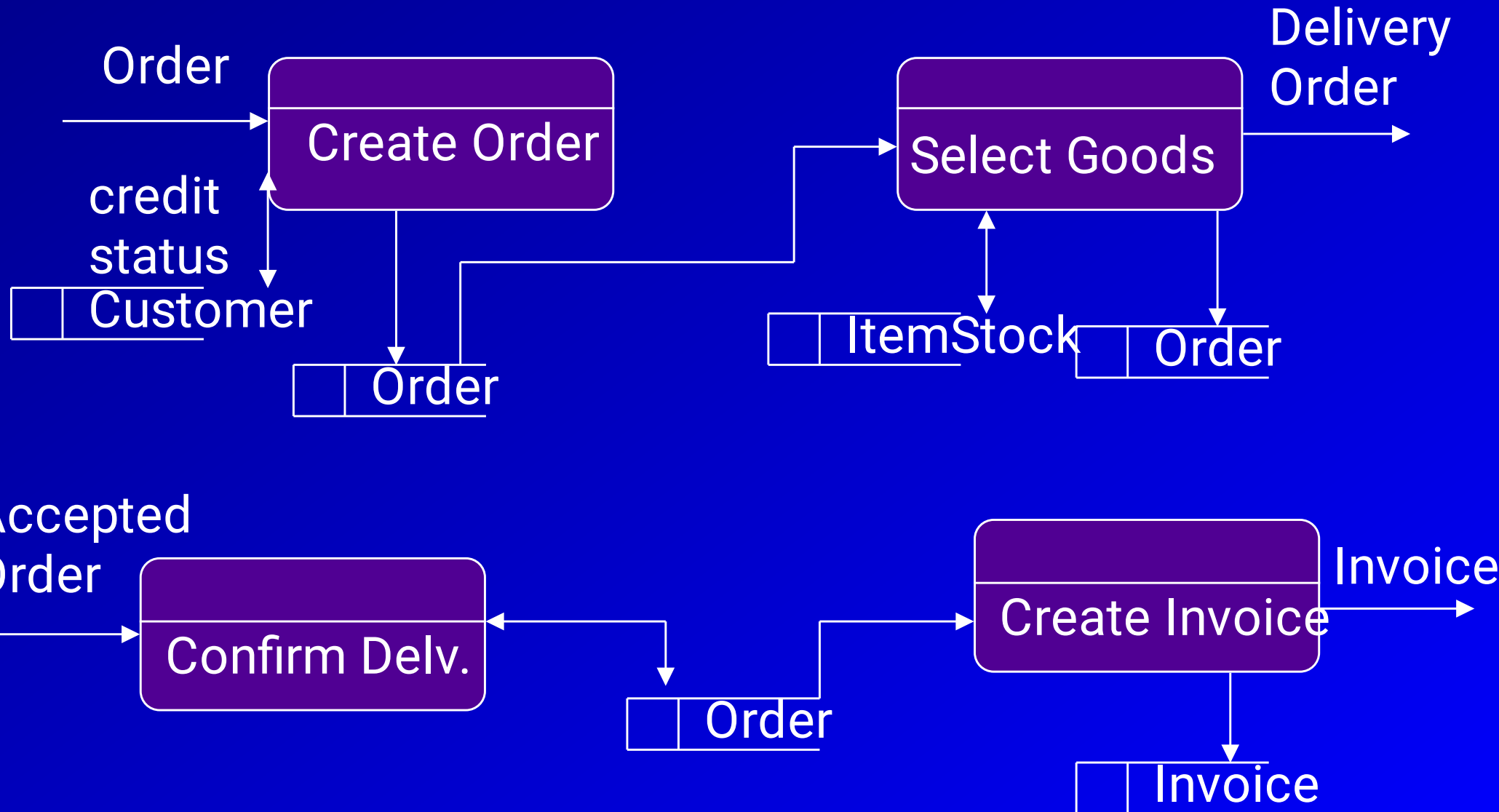
Process Description Tools

F Decision Tables

- Can have more than two possible outcomes
- Often are the best way to describe a complex set of conditions

	1	2	3	4	5	6	7	8
Preferred customer	Y	Y	Y	Y	N	N	N	N
Ordered more than \$1,000	Y	Y	N	N	Y	Y	N	N
Used our charge card	Y	N	Y	N	Y	N	Y	N
5% discount	X	X						
Additional 5% discount	X							
\$25 bonus coupon			X					
\$5 bonus coupon				X	X	X	X	X

Example: Current Logical DFD (online shopping store Level 2)



Process Description Tools

F Decision Tables

- Shows a logical structure, with all possible combinations of conditions and resulting actions
- It is important to consider every possible outcome to ensure that you have overlooked nothing

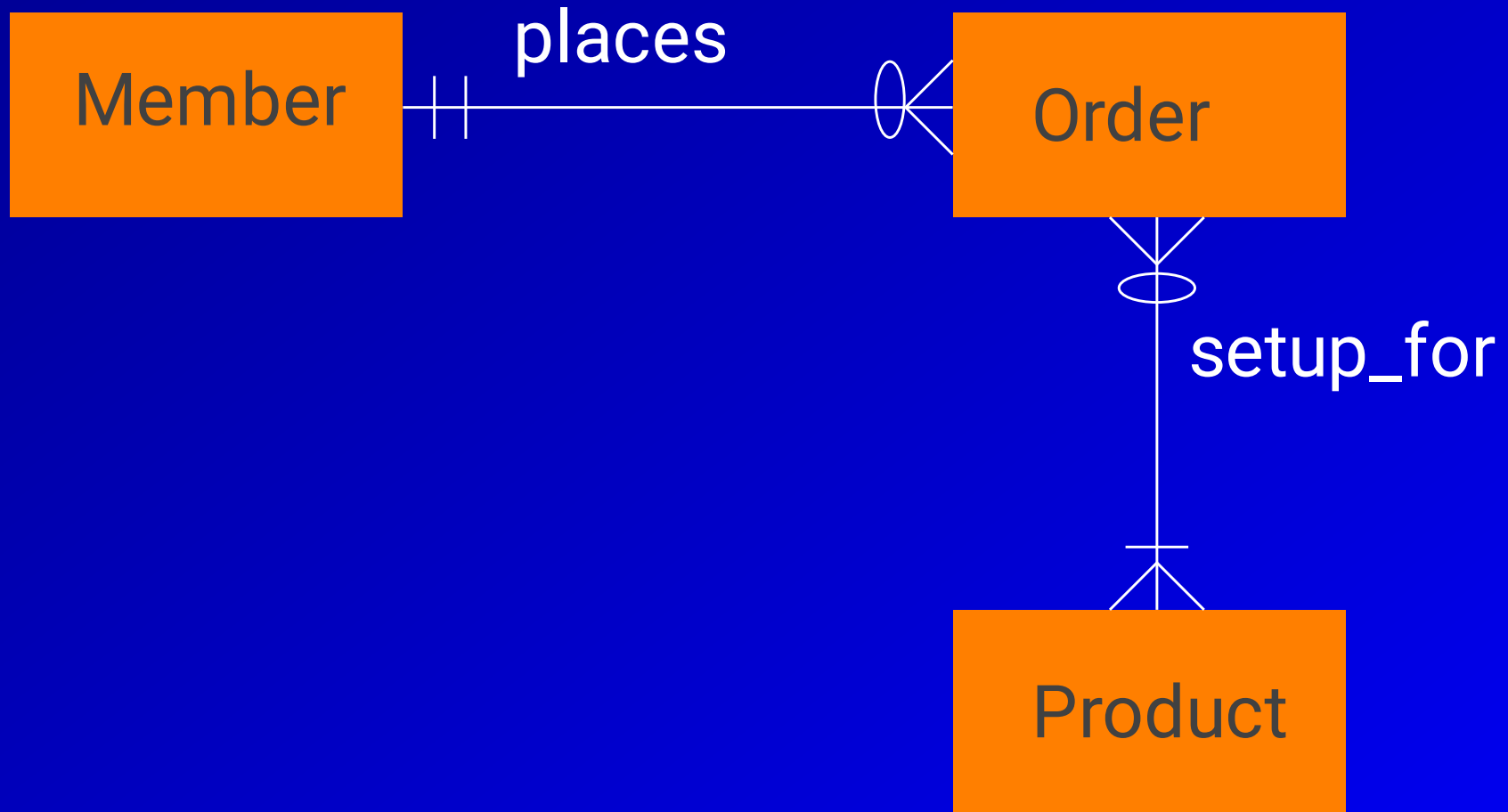
	1	2	3	4
Credit status is OK	Y	Y	N	N
Product is in stock	Y	N	Y	N
Accept order	X			
NO Reject order		X	X	X

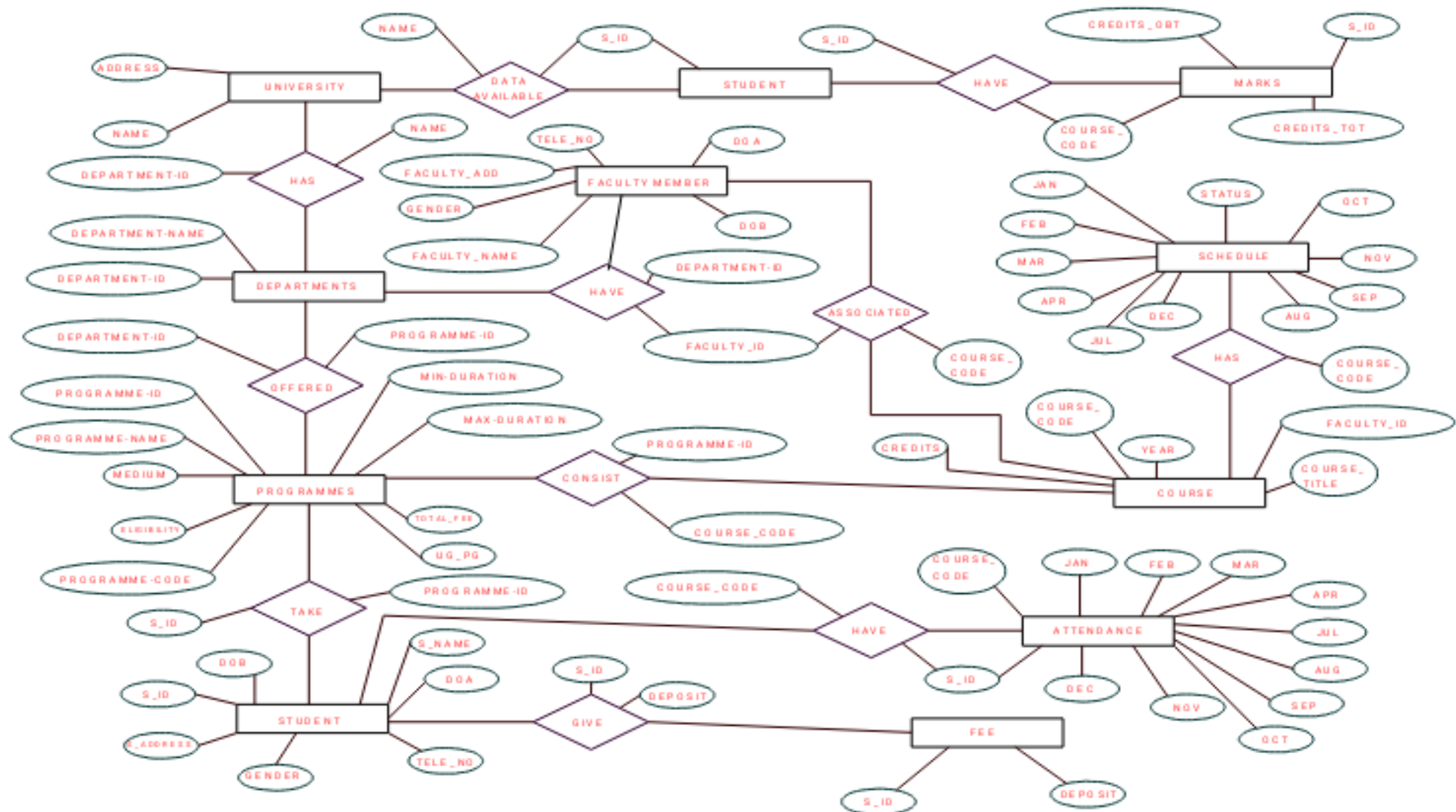
Data Modeling

Entity Relationship Diagram (ERD)

- F It gives a logical view of the data and the relationships between them in a system.
- F Major components:
 - Entities
 - Relationships between entities
 - Cardinality
 - one-to-one recursive relationship
 - one-to-many super type -subtype
 - many-to-many

Example: ERD





The Data Dictionary

- F The data dictionary is an organized listing of all data elements pertinent to the system with precise rigorous definitions so that both user and SA will have common understanding of all data elements.
- F It describes:
 - meaning and composition of data flows, data stores
 - relevant values and units of elementary data elements
 - details of relationships between stores that are highlighted in a ERD

Data Dictionary Notation

Example:

customer-name = title + first-name +
(middle-name) + last-name

title = [Mr. | Miss | Ms. | Mrs. | Dr. | Prof.]

first-name = {legal-character}

order = customer-name + shipping-addr
+ {item}

D D...

=	is composed of
+	and
[]	either / or
{ }	repetition of (string)
()	optional data
---	for comments

Modeling Time-Dependent System Behavior

- F State Transition Diagram
- F Entity Life Histories

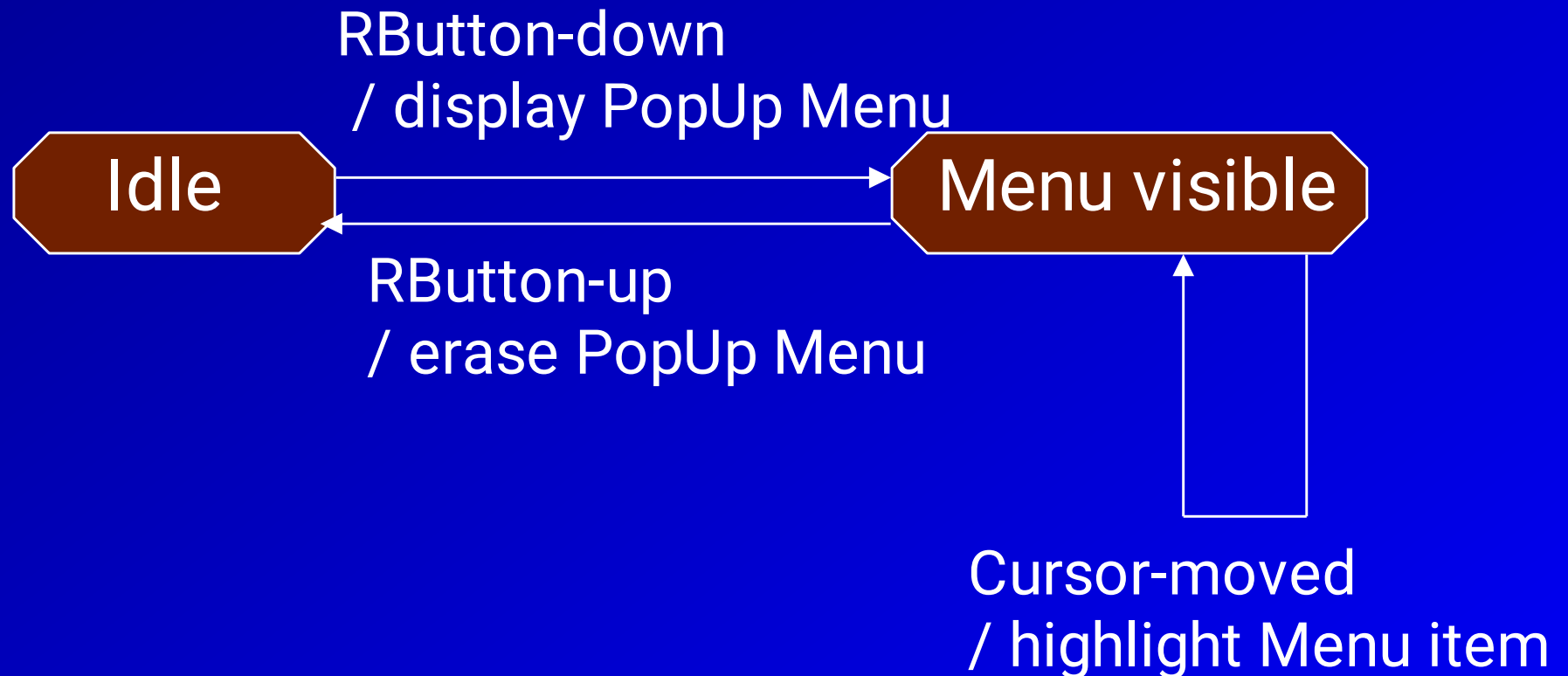
State Transition Diagram (STD)

- F It highlights the time-dependent behavior of a system.
- F It shows 'what' happens 'when'
- F Major components
 - States
 - Transitions between states
- F Useful for describing behavior of
 - real-time systems, interactive systems

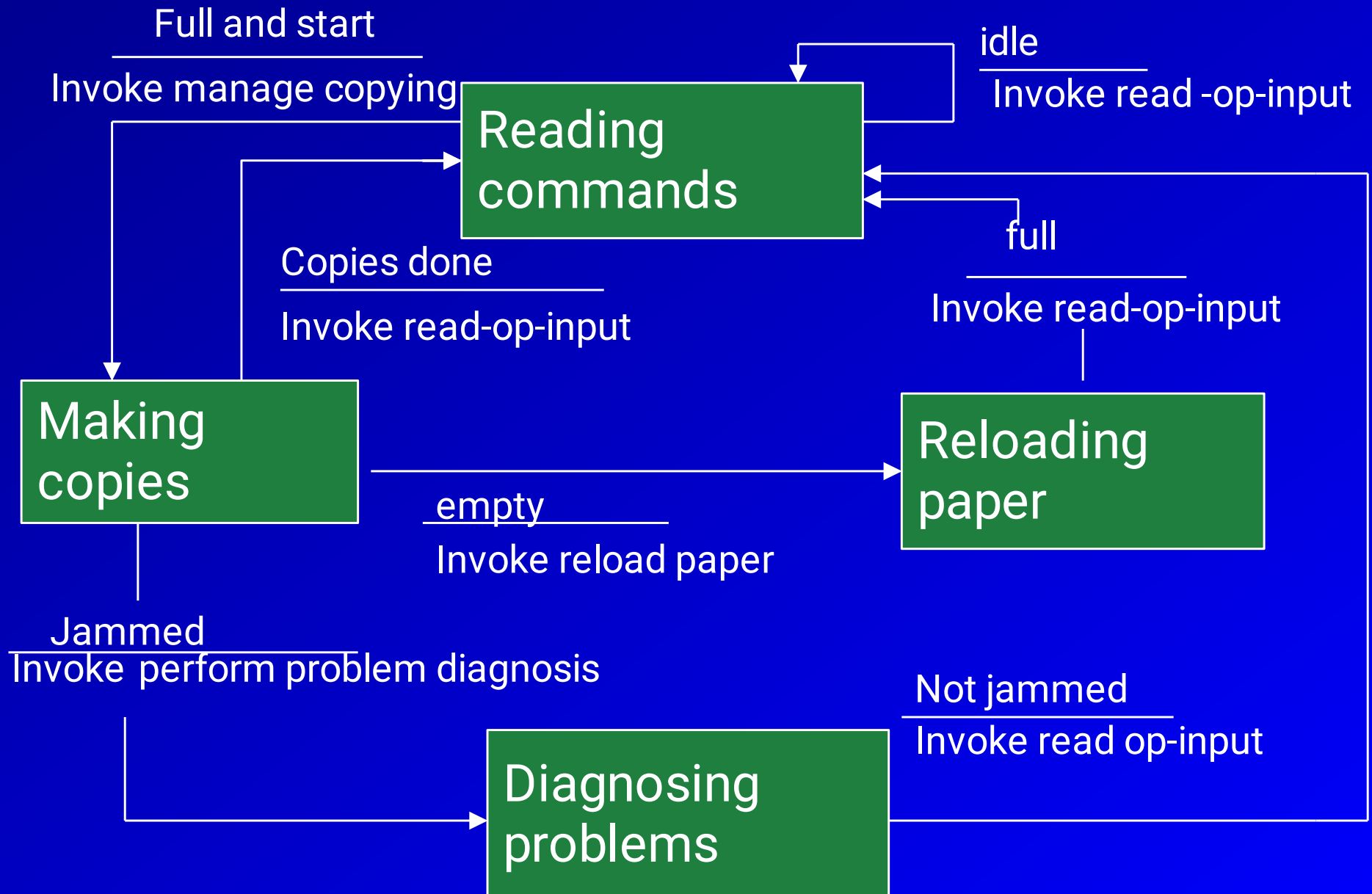
State Transition Diagram...

- F The state transition diagram (std) represents the behavior of a system by displaying it's states and the events which cause system to change its state.
- F STD indicates what actions are taken as a consequence of a particular event.

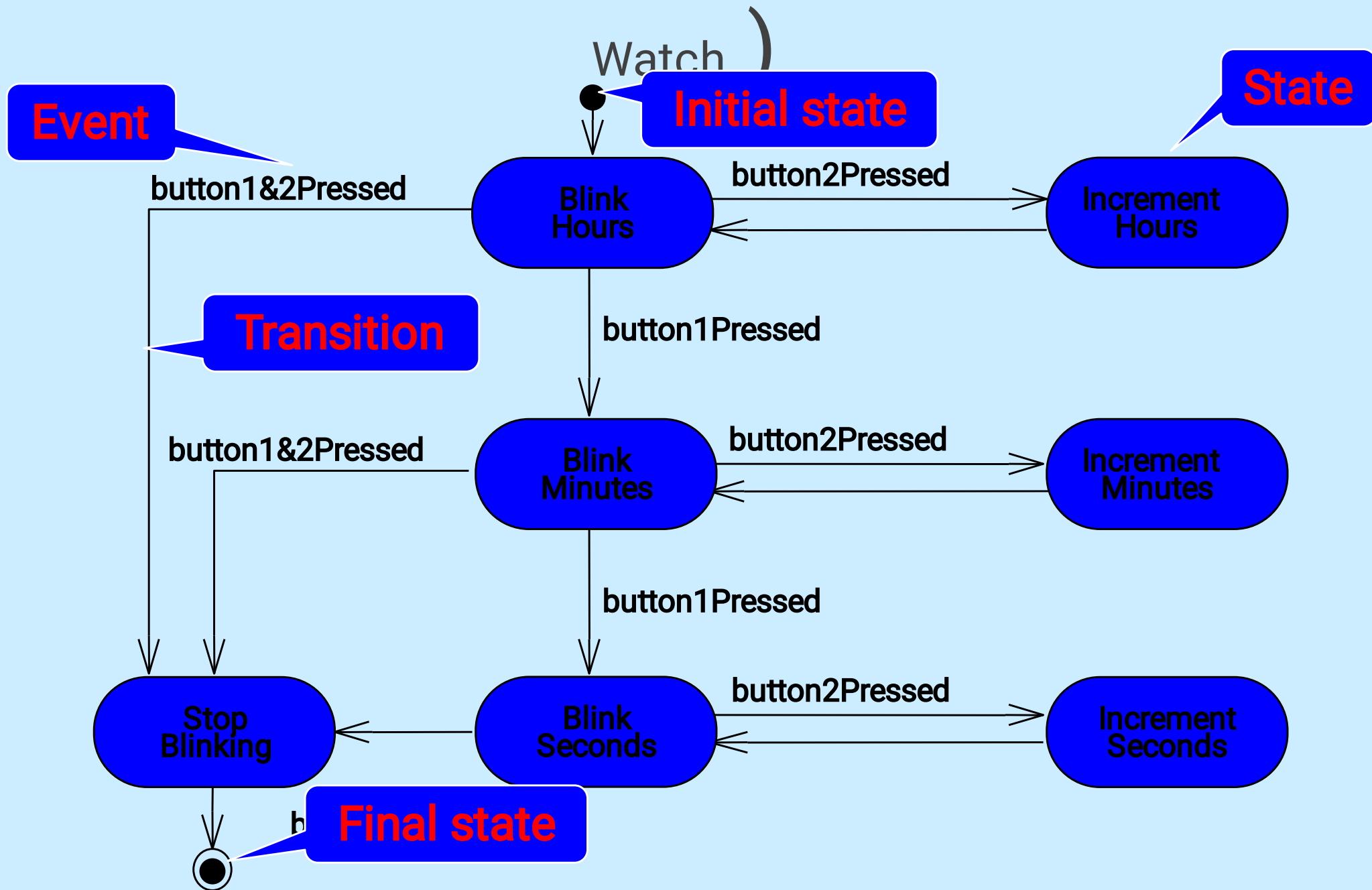
Example: State Transition Diagram



STD for photocopier software



Statechart Diagrams (ex. Digital



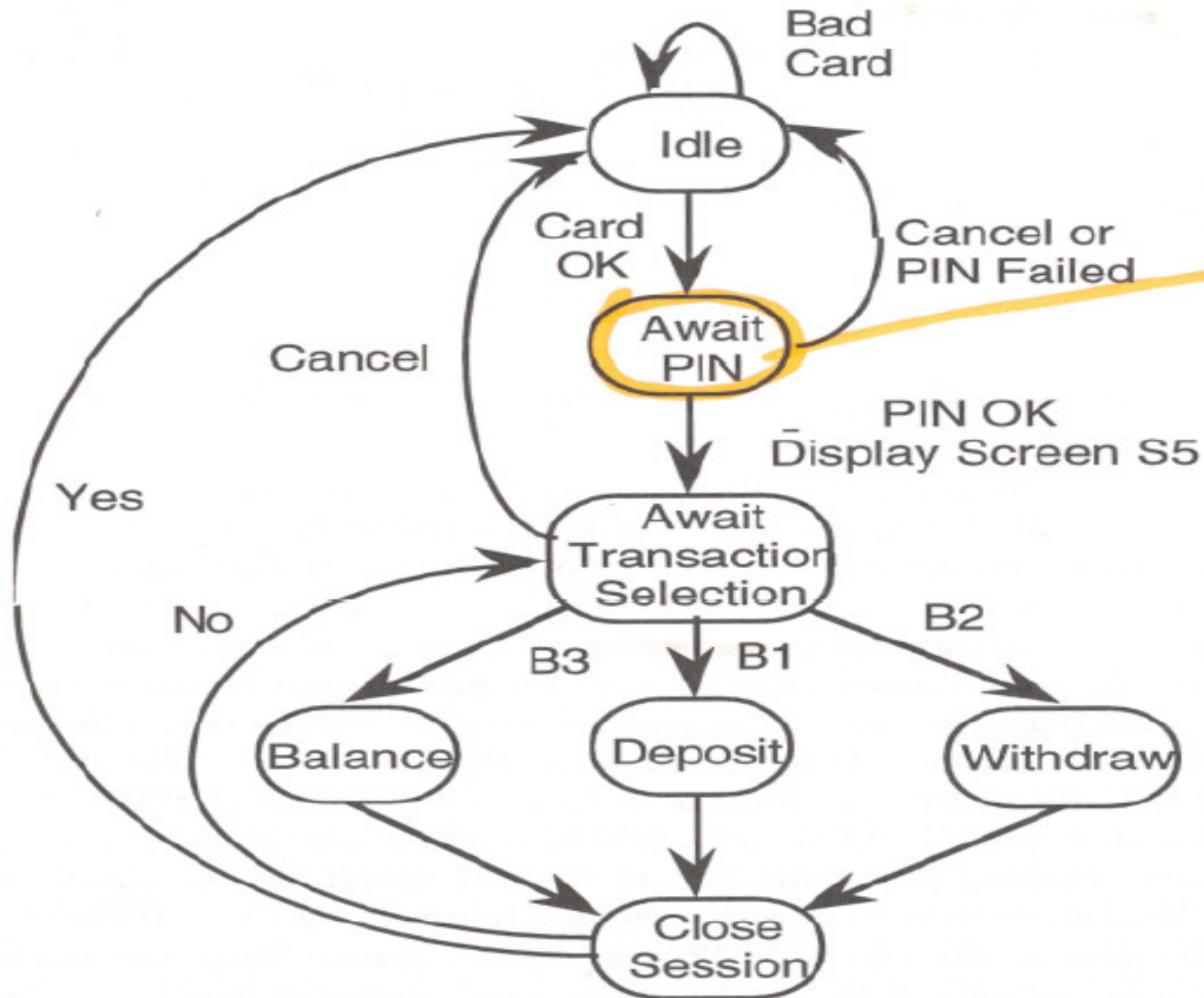


Figure 12.12 Upper Level SATM Finite State Machine

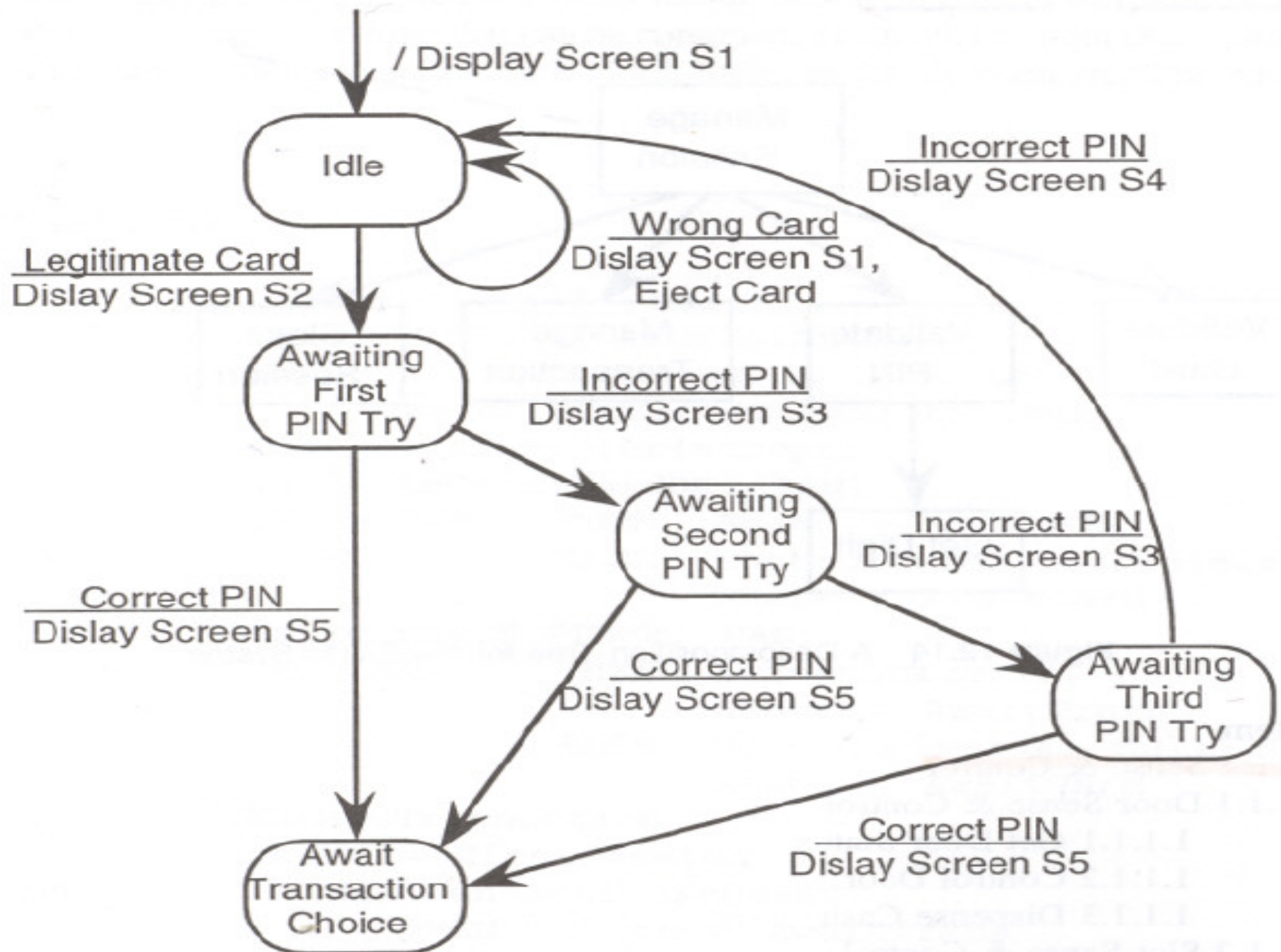
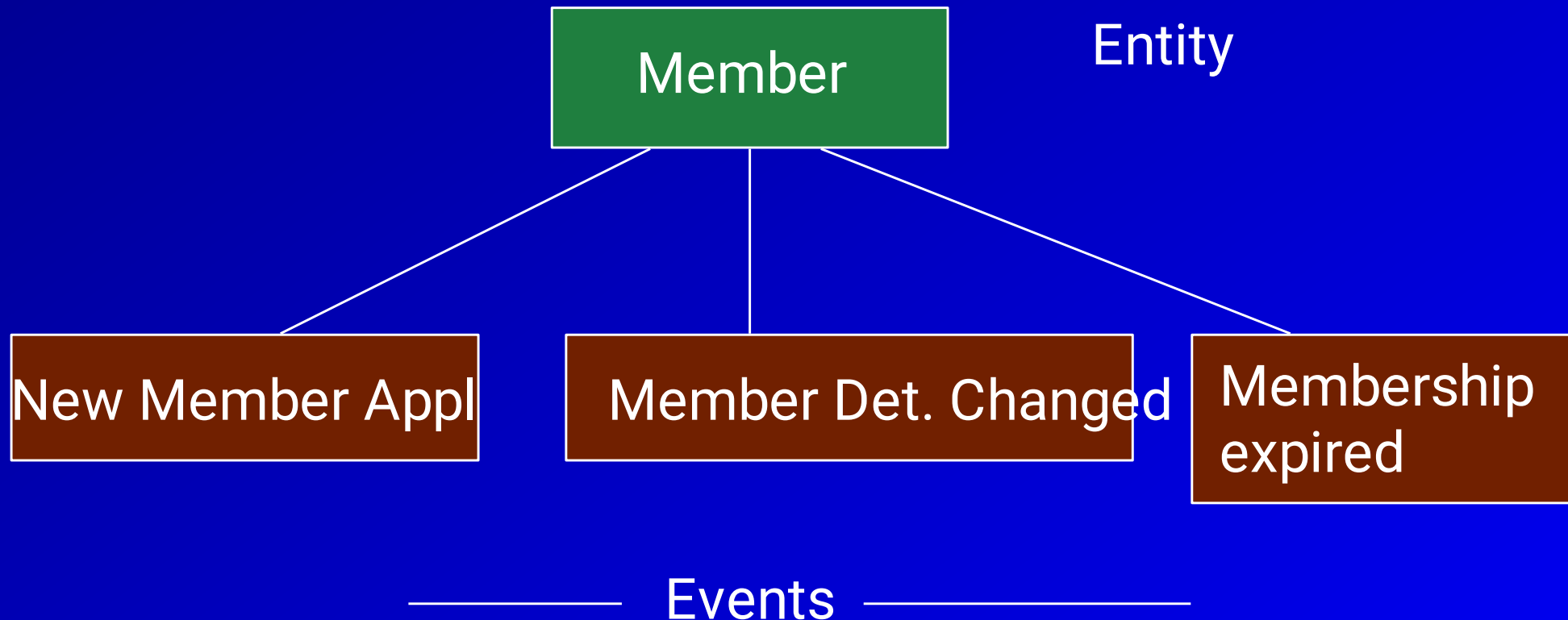


Figure 12.13 PIN Entry Finite State Machine

Entity Life History (ELH)

- F An ELH represents all the permitted sequences of events that may occur during the life of an occurrence of an entity.
- F An Event may effect entity occurrences
- F An effect can be to:
 - Create, Modify, Delete
- F To construct ELH, first construct Entity/Event Matrix

Example: Entity Life History



Cross Checking the different Models

- F Balancing ERD against the DFD, Process specs.
- F Balancing DFD against the STD
- F Balancing ERD against Data dictionary
- F Balancing DFD against Data dictionary

Balancing ERD and DFD, Process Specs.

- F Every store on the DFD must correspond to an entity or a relationship or a combination of an entity and relationship on the ERD.
- F Entity names on the ERD and data store names on the DFD must match.
- F The data dictionary entries must apply to both the DFD and the ERD data elements.

Function Based Metrics (check pints) for analysis model

1. Needs reliable backup/recovery?
2. Needs data communications?
3. Are there distributed processing functions?
4. Is performance critical?
5. System runs in existing heavily utilized operational environment?
6. System requires on-line data entry?
7. On-line entry requires I/p transaction over multiple screens/operations?

Continue...

8. Are master files updated on line?
9. Are I/Ps ,O/Ps , files or enquiries complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion & installation included in the design?
13. Is system designed for multiple installations in diff. Organizations?
14. Is application designed to facilitate change & ease of use by the user?

Function-Based Metrics for analysis model

Quantitative representation

- F Scale 0 – 5
(not important/applicable) - (absolutely essential)
- F Errors/FP
- F Defects/FP
- F \$ (rate)/FP
- F Pages of documentation/FP
- F FP/person-month

The Basic Metric for analysis model

- Functional Primitives (FuP)
No. of transformations on lowest level DFD
- Data Elements (DE)
No. of attributes of a data object
- Objects (OB)
No. of data objects
- Relationships (RE)
No. of connections between data objects
- States (ST)
No. of user observable states in STD
- Transitions (TR)
No. of state transitions in STD