
Kendall & Kendall
Systems Analysis and Design, 9e

Using Data Flow Diagrams

Learning Objectives

1. Comprehend the importance of using logical and physical data flow diagrams (DFDs) to graphically depict movement for humans and systems in an organization.
2. Create, use, and explode logical DFDs to capture and analyze the current system through parent and child levels.
3. Develop and explode logical DFDs that illustrate the proposed system.
4. Produce physical DFDs based on logical DFDs you have developed.
5. Understand and apply the concept of partitioning of physical DFDs.



Major Topics

1. Data flow diagram symbols
2. Data flow diagram levels
3. Creating data flow diagrams
4. Physical and logical data flow diagrams
5. Partitioning
6. Communicating using data flow diagrams



Data Flow Diagrams


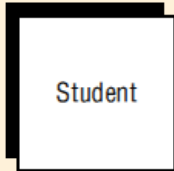



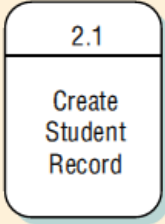

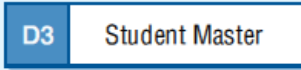
- Graphically characterize data processes and flows in a business system
- Depict:
 - System inputs
 - Processes
 - Outputs



Basic Symbols

1. A (double) square: an external entity
2. An arrow: a movement of data from one point to another
3. A rectangle with rounded corners: a transforming process
4. An open-ended rectangle: a data store

The Four Basic Symbols Used in Data Flow Diagrams, Their Meanings, and Examples (Figure 7.1)

Symbol	Meaning	Example
	Entity	
	Data Flow	
	Process	
	Data Store	

External Entities

- A **source** or **destination** of data, outside the boundaries of the system
- Represent another department, a business, a person, or a machine
- Should be named with a noun

Group Discussion

- ❑ Number of members per group: 3-5 students
- ❑ Time: 10 mins.
- ❑ Find the source(s) and destination(s) of
 1. The Selling process of a supermarket.
 2. The Course registration process of an university.



Data Flow

- Shows movement of data from one point to another
- Described with a noun
- Arrowhead indicates the flow direction
- Represents data about a person, place, or thing

Process

- Denotes a change in or transformation of data
- Represents work being performed in the system
- Naming convention:
 - Assign the name of the whole system when naming a high-level process
 - To name a major subsystem attach the word subsystem to the name
 - Use the form verb-adjective-noun for detailed processes

Process

- Naming convention example:
 - To name of the whole system (a high-level process):
 - INVENTORY CONTROL SYSTEM
 - To name a major subsystem:
 - INVENTORY REPORTING SUBSYSTEM
 - INTERNET CUSTOMER FULFILLMENT SYSTEM
 - Use the form verb-adjective-noun for detailed processes
 - COMPUTE SALES TAX,
 - VERIFY CUSTOMER ACCOUNT STATUS,
 - PREPARE SHIPPING INVOICE,
 - SEND CUSTOMER EMAIL CONFIRMATION,
 - ADD INVENTORYRE RECORD,
 - COMPUTE CUSTOMER'S DISCOUNT

Group Discussion

- ❑ Number of members per group: 3-5 students
- ❑ Time: 10 mins.
- ❑ Using the situation of previous group discussion, name of
 1. The whole system (high level)
 2. The subsystems (if it has)
 3. At least 3 detailed processes

Data Store

- A depository for data that allows examination, addition, and retrieval of data
- Named with a noun, describing the data
- Data stores are usually given a unique reference number, such as D1, D2, D3
- Represents a:
 - Database
 - Computerized file
 - Filing cabinet

Data Store

- Examples:
 - Customer
 - Product
 - Order
 - OrderList
 - Employee
 - ...

Steps in Developing Data Flow Diagrams

1. Make **a list of business activities** and use it to determine various: (1) External entities, (2) Data flows, (3) Processes, (4) Data stores
2. Create a context diagram that shows external entities and data flows to and from the system. **Do not show any detailed processes or data stores.**
3. Draw Diagram 0, the next level. Show processes, but keep them general. Show data stores at this level.
4. Create a child diagram for each of the processes in Diagram 0.
5. Check for errors and make sure the labels you assign to each process and data flow are meaningful.
6. Develop a physical data flow diagram from the logical data flow diagram. **Distinguish between manual and automated processes**, describe actual files and reports by name, and add controls to indicate when processes are complete or errors occur.
7. Partition the physical data flow diagram by separating or grouping parts of the diagram in order to facilitate programming and implementation.

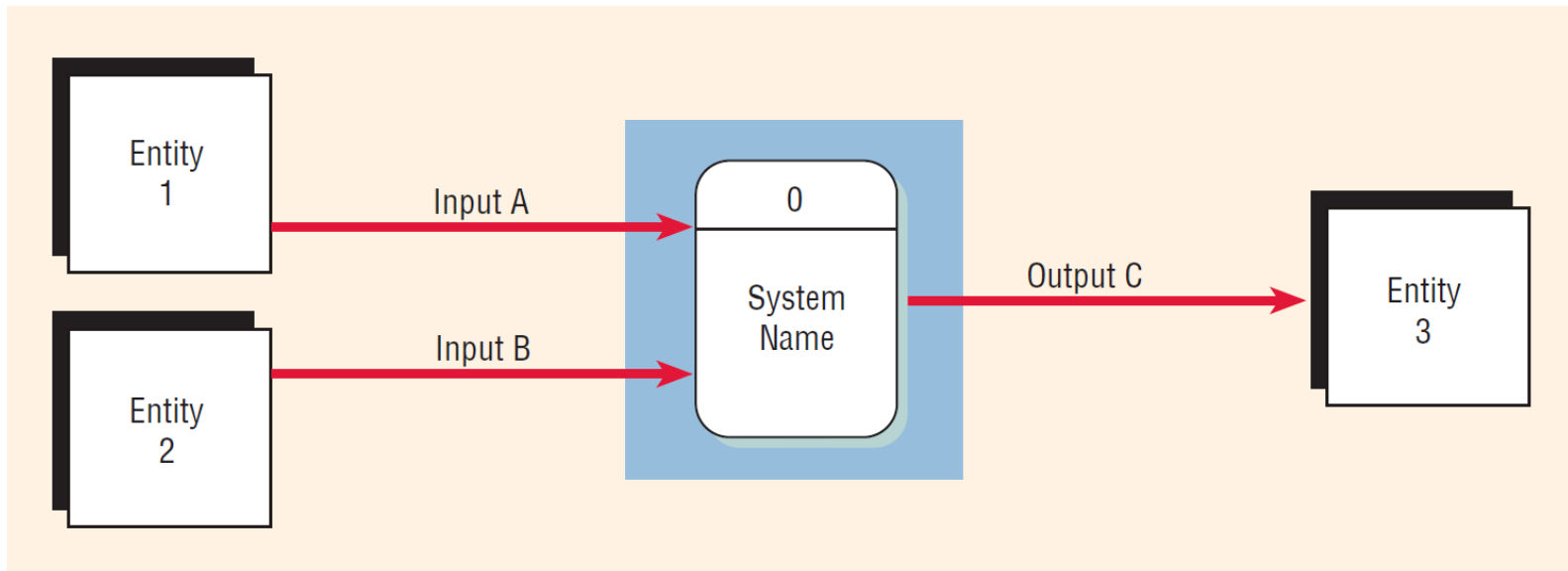
Creating the Context Diagram

- The highest level in a data flow diagram
- Contains only one process, representing the entire system
- The process is given the number 0
- All external entities, as well as major data flows are shown

Basic Rules

- The data flow diagram must have one process
- Must not be any freestanding objects
- A process must have both an input and output data flow
- A data store must be connected to at least one process
- External entities should not be connected to one another

Context Diagram (Figure 7.3)



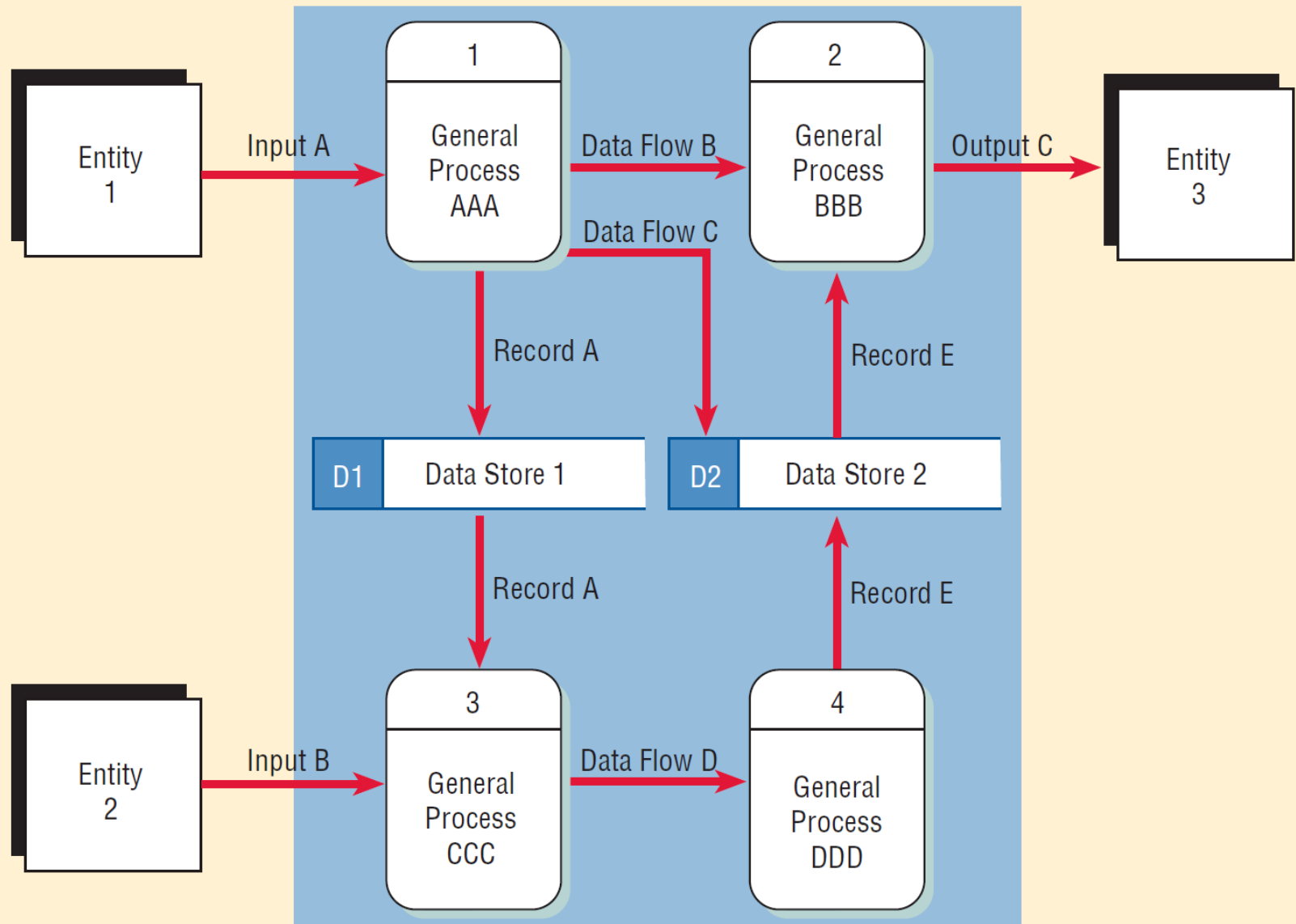
Drawing Diagram 0

- The explosion of the context diagram
- May include up to **nine** processes
- Each process is numbered
- Major data stores and all external entities are included

Drawing Diagram 0 (continued)

- Start with the data flow from an entity on the input side
- Work backward from an output data flow
- Examine the data flow to or from a data store
- Analyze a well-defined process
- **Take note of any fuzzy areas**

Note Greater Detail in Diagram 0 (Figure 7.3)



Data Flow Diagram Levels

- Data flow diagrams are built in layers
- The top level is the context level
- Each process may explode to a lower level
- The lower level diagram number is the same as the parent process number
- Processes that do not create a child diagram are called primitive

Creating Child Diagrams

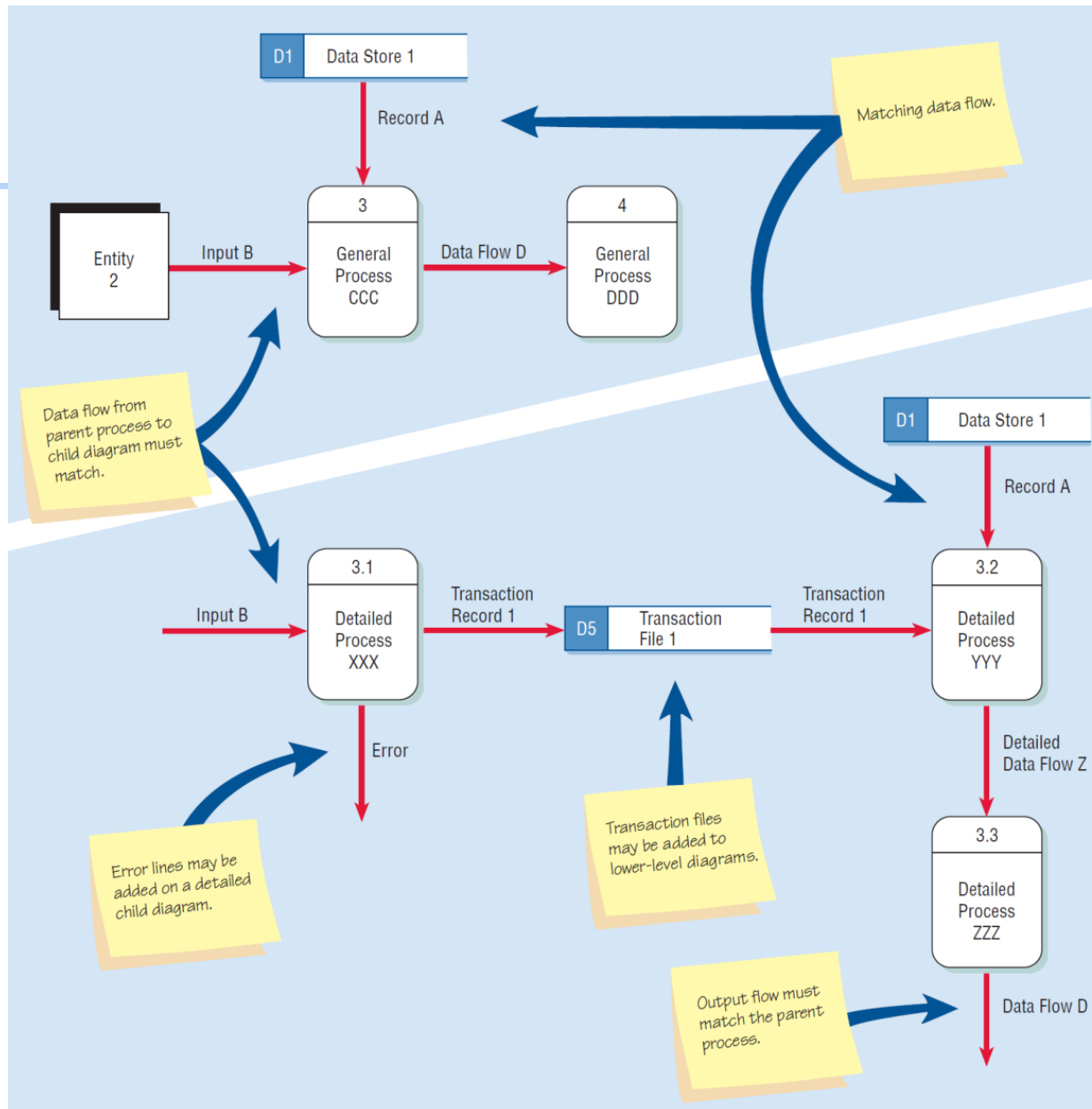
- Each process on diagram 0 may be exploded to create a child diagram
- A child diagram cannot produce output or receive input that the parent process does not also produce or receive
- The child process is given the same number as the parent process
 - Process 3 would explode to Diagram 3



Creating Child Diagrams (continued)

- Entities are usually not shown on the child diagrams below Diagram 0
- If the parent process has data flow connecting to a data store, the child diagram may include the data store as well
- When a process is not exploded, it is called a primitive process

Differences between the Parent Diagram (above) and the Child Diagram (below) (Figure 7.4)





Data Flow Diagrams Error Summary

- Forgetting to include a data flow or pointing an arrow in the wrong direction
- Connecting data stores and external entities directly to each other
- Incorrectly labeling processes or data flow

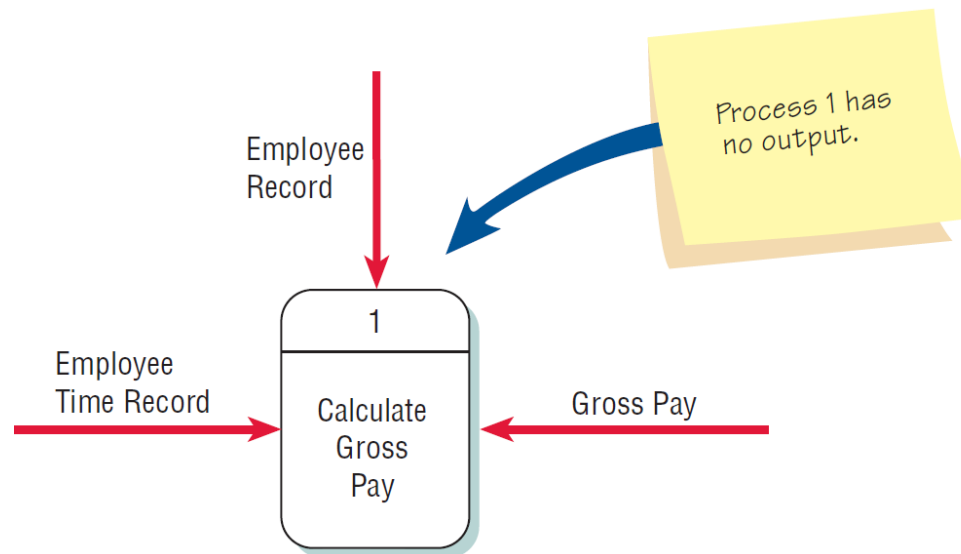


Data Flow Diagrams Error Summary (continued)

- Including more than nine processes on a data flow diagram
- Omitting data flow
- Creating unbalanced decomposition (or explosion) in child diagrams

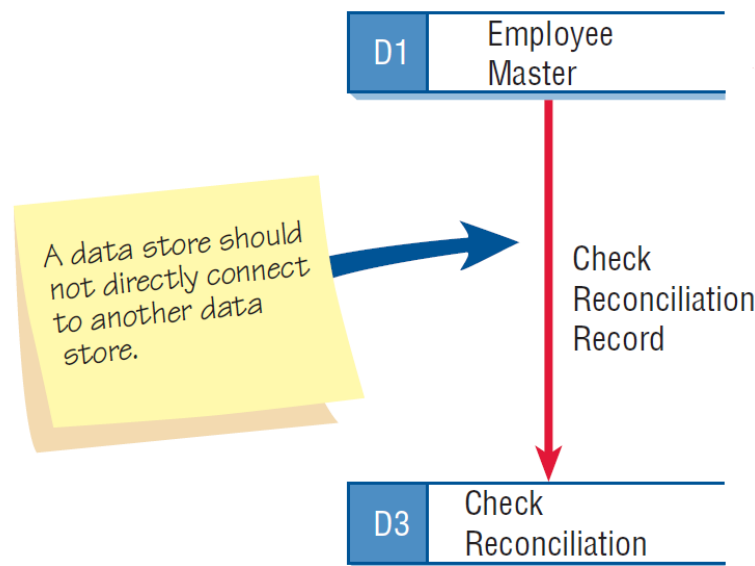
Checking the Diagrams for Errors (Figure 7.5)

- Forgetting to include a data flow or pointing an arrow in the wrong direction

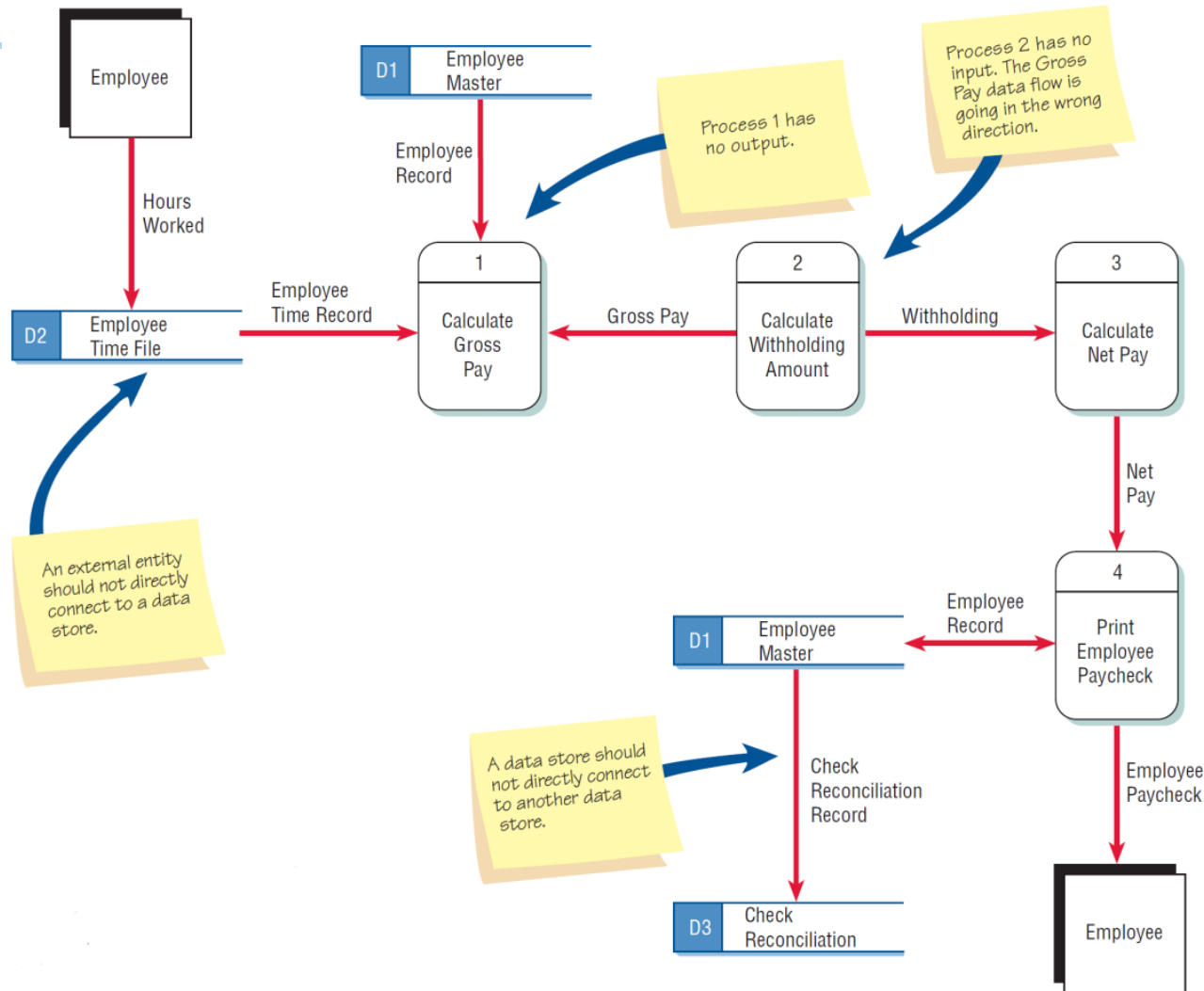


Checking the Diagrams for Errors (continued Figure 7.5)

- Connecting data stores and external entities directly to each other



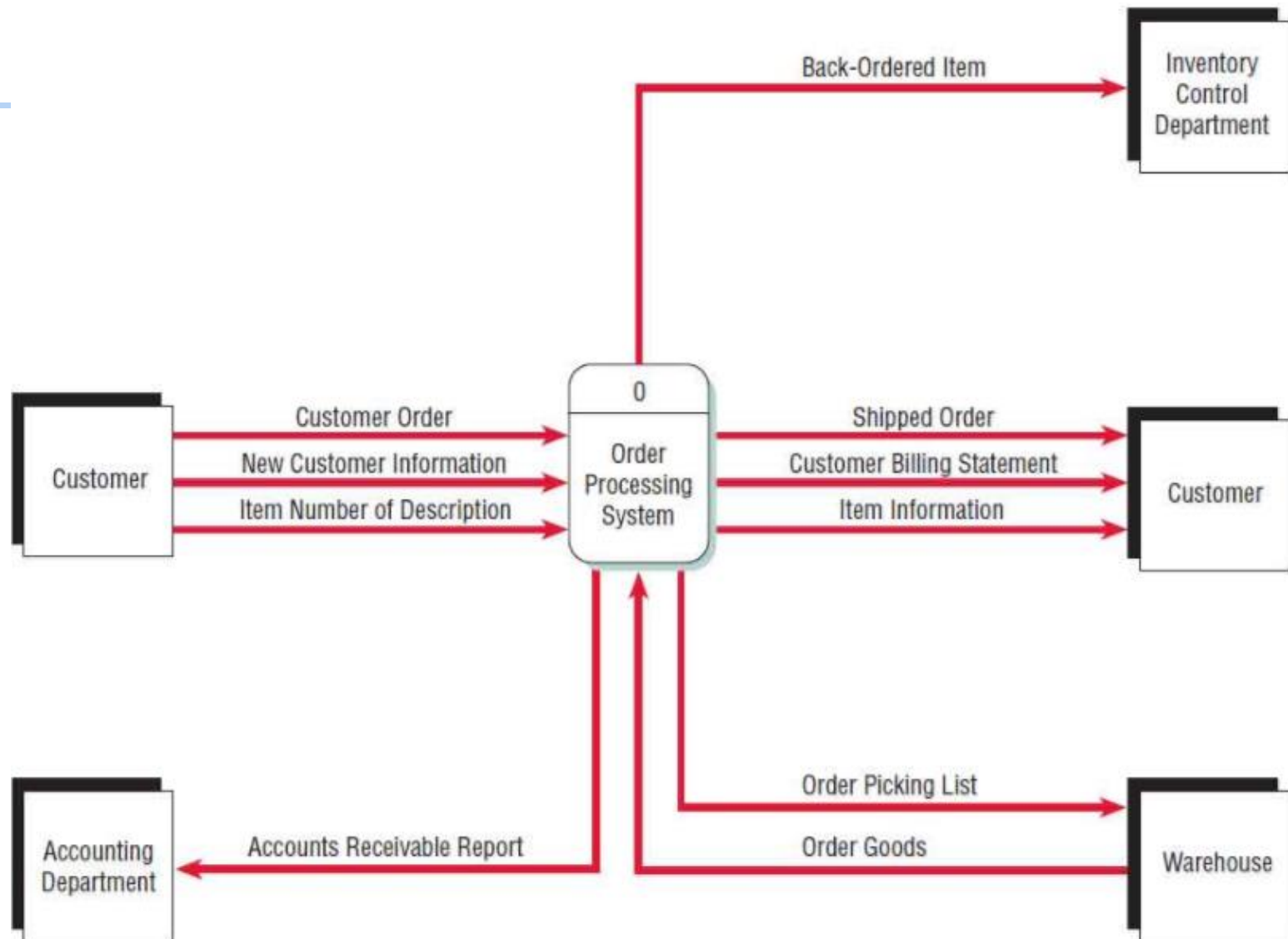
Typical Errors that Can Occur in a Data Flow Diagram (Payroll Example) (continued Figure 7.5)



Group Discussion

- ❑ Number of members per group: 5 students
- ❑ Time: 10 mins.
- ❑ Produce the **Context-level DFD** for Business Activities

Context-level DFD



Group Discussion

- ❑ 2 groups
- ❑ Time: 20 mins.
- ❑ Produce the **DFD 0** based on the **Context-level DFD** above.

Diagram 0, of the order processing system

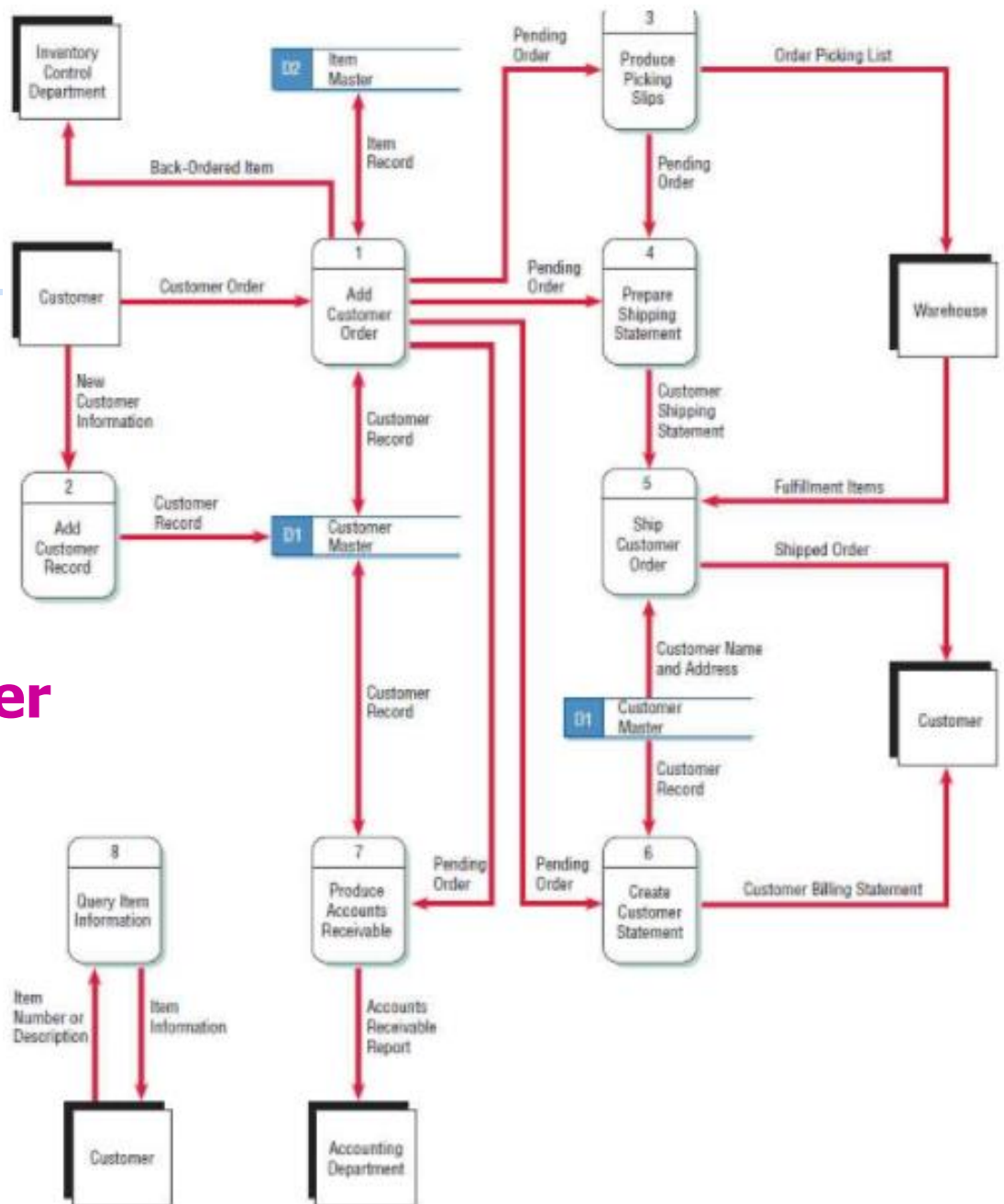
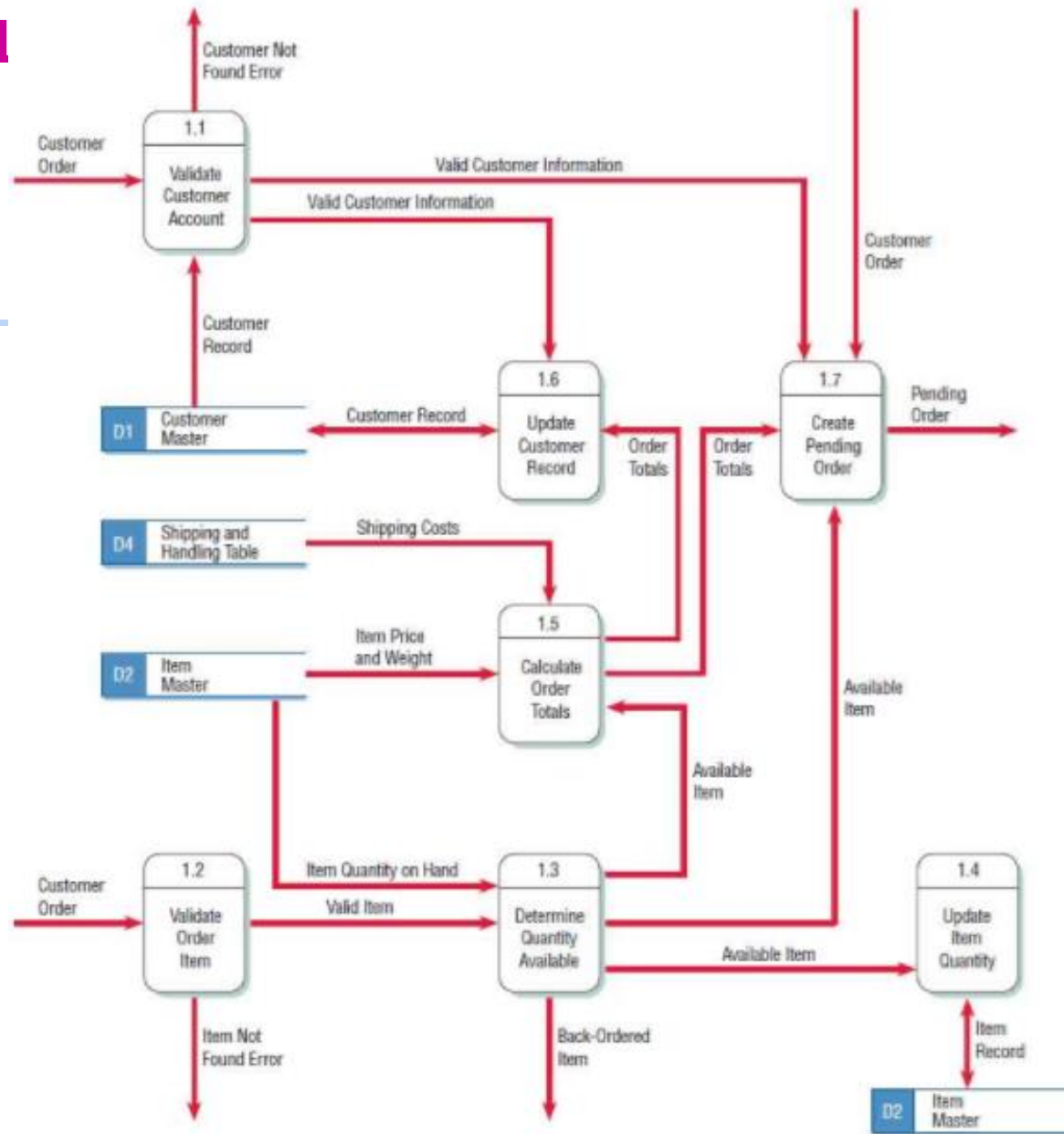


Diagram 1





Logical and Physical Data Flow Diagrams

- Logical
 - Focuses on the business and how the business operates
 - Not concerned with how the system will be constructed
 - Describes the business events that take place and the data required and produced by each event



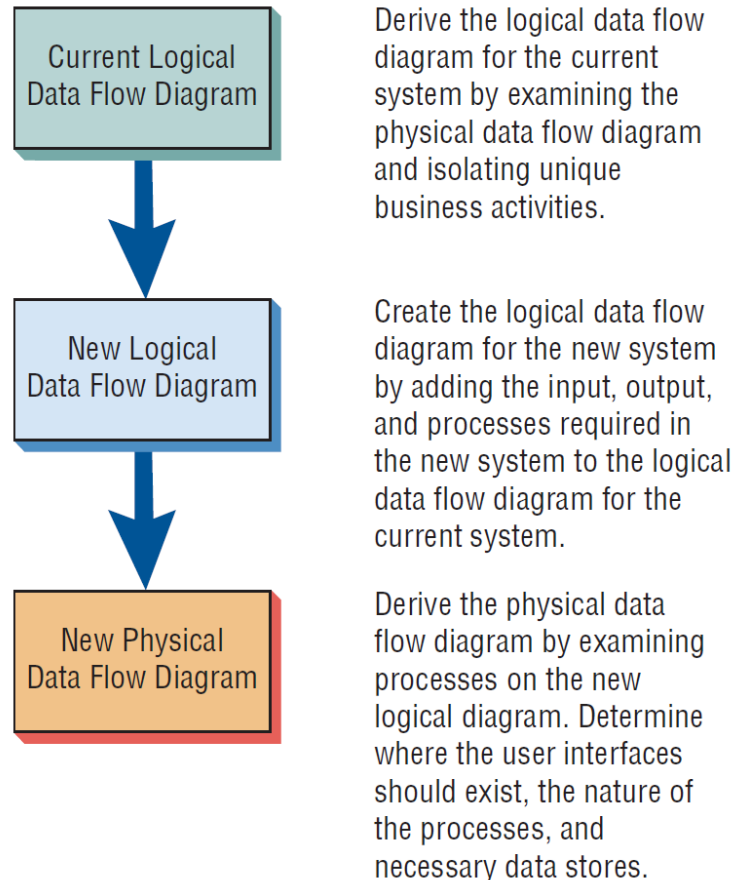
Logical and Physical Data Flow Diagrams

- Physical
 - Shows how the system will be implemented
 - Depicts the system

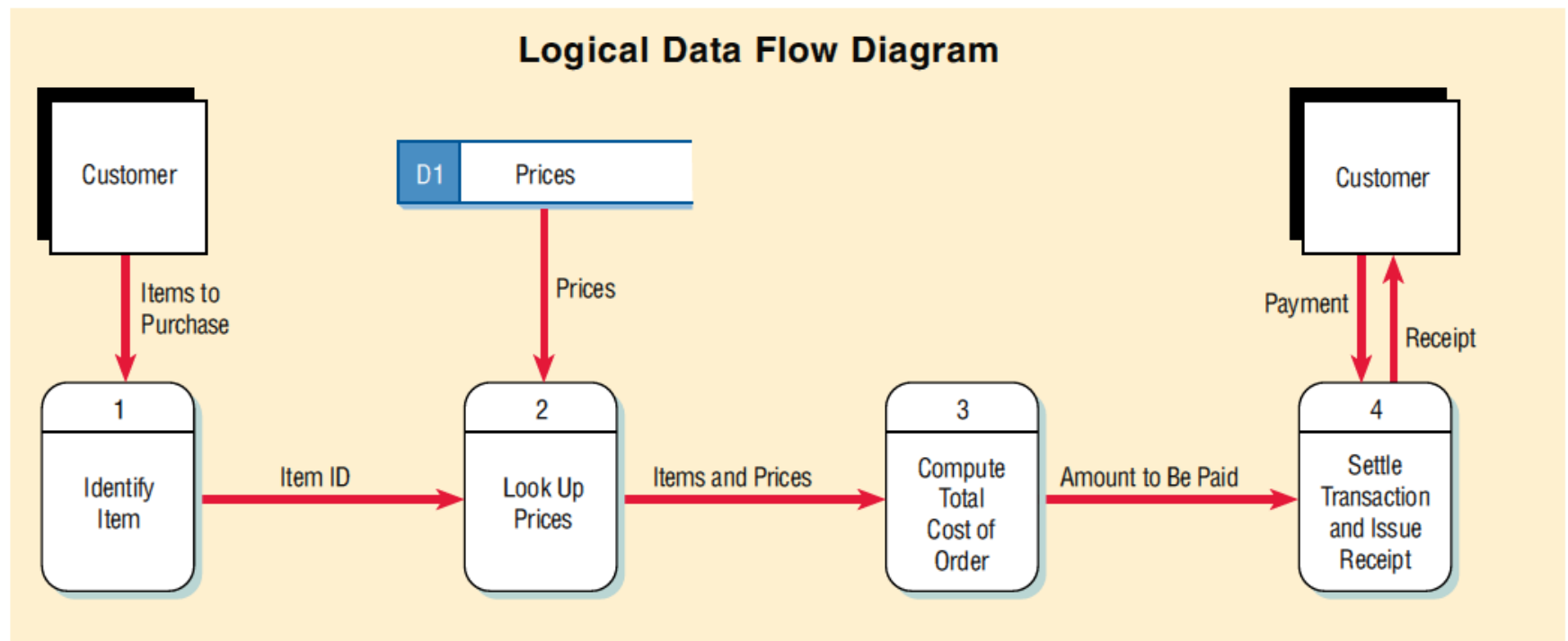
Features Common of Logical and Physical Data Flow Diagrams (Figure 7.7)

Design Feature	Logical	Physical
What the model depicts	How the business operates.	How the system will be implemented (or how the current system operates).
What the processes represent	Business activities.	Programs, program modules, and manual procedures.
What the data stores represent	Collections of data regardless of how the data are stored.	Physical files and databases, manual files.
Type of data stores	Show data stores representing permanent data collections.	Master files, transition files. Any processes that operate at two different times must be connected by a data store.
System controls	Show business controls.	Show controls for validating input data, for obtaining a record (record found status), for ensuring successful completion of a process, and for system security (example: journal records).

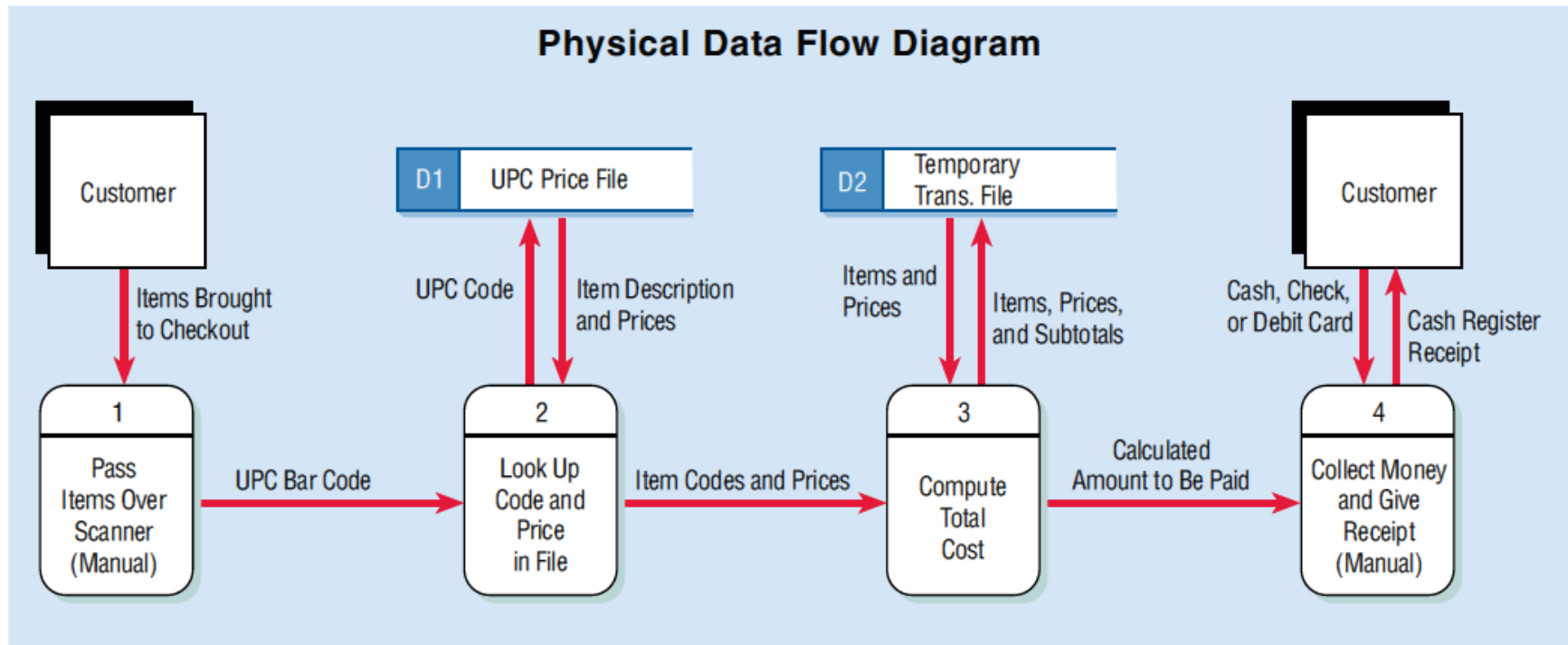
The Progression of Models from Logical to Physical (Figure 7.8)



Logical Data Flow Diagram Example (Figure 7.9)



Physical Data Flow Diagram Example (Figure 7.9)





Developing Logical Data Flow Diagrams

- Better communication with users
- More stable systems
- Better understanding of the business by analysts
- Flexibility and maintenance
- Elimination of redundancy and easier creation of the physical model



Developing Physical Data Flow Diagrams

- Clarifying which processes are performed by humans and which are automated
- Describing processes in more detail
- Sequencing processes that have to be done in a particular order
- Identifying temporary data stores
- Specifying actual names of files and printouts
- Adding controls to ensure the processes are done properly

Physical Data Flow Diagrams Contain Many Items Not Found in Logical Data Flow Diagrams (Figure 7.10)

Contents of Physical Data Flow Diagrams

- Manual processes
- Processes for adding, deleting, changing, and updating records
- Data entry and verifying processes
- Validation processes for ensuring accurate data input
- Sequencing processes to rearrange the order of records
- Processes to produce every unique system output
- Intermediate data stores
- Actual file names used to store data
- Controls to signify completion of tasks or error conditions

CRUD Matrix

- The acronym CRUD is often used for
 - Create
 - Read
 - Update
 - Delete
- These are the activities that must be present in a system for each master file
- A CRUD matrix is a tool to represent where each of these processes occurs in a system

CRUD Matrix (Figure 7.11)

Activity	Customer	Item	Order	Order Detail
Customer Logon	R			
Item Inquiry		R		
Item Selection		R	C	C
Order Checkout	U	U	U	R
Add Account	C			
Add Item		C		
Close Customer Account	D			
Remove Obsolete Item		D		
Change Customer Demographics	RU			
Change Customer Order	RU	RU	RU	CRUD
Order Inquiry	R	R	R	R



Use Cases and Data Flow Diagrams

- Each use case defines one activity and its trigger, input, and output
- Allows the analyst to work with users to understand the nature of the processes and activities and then create a single data flow diagram fragment

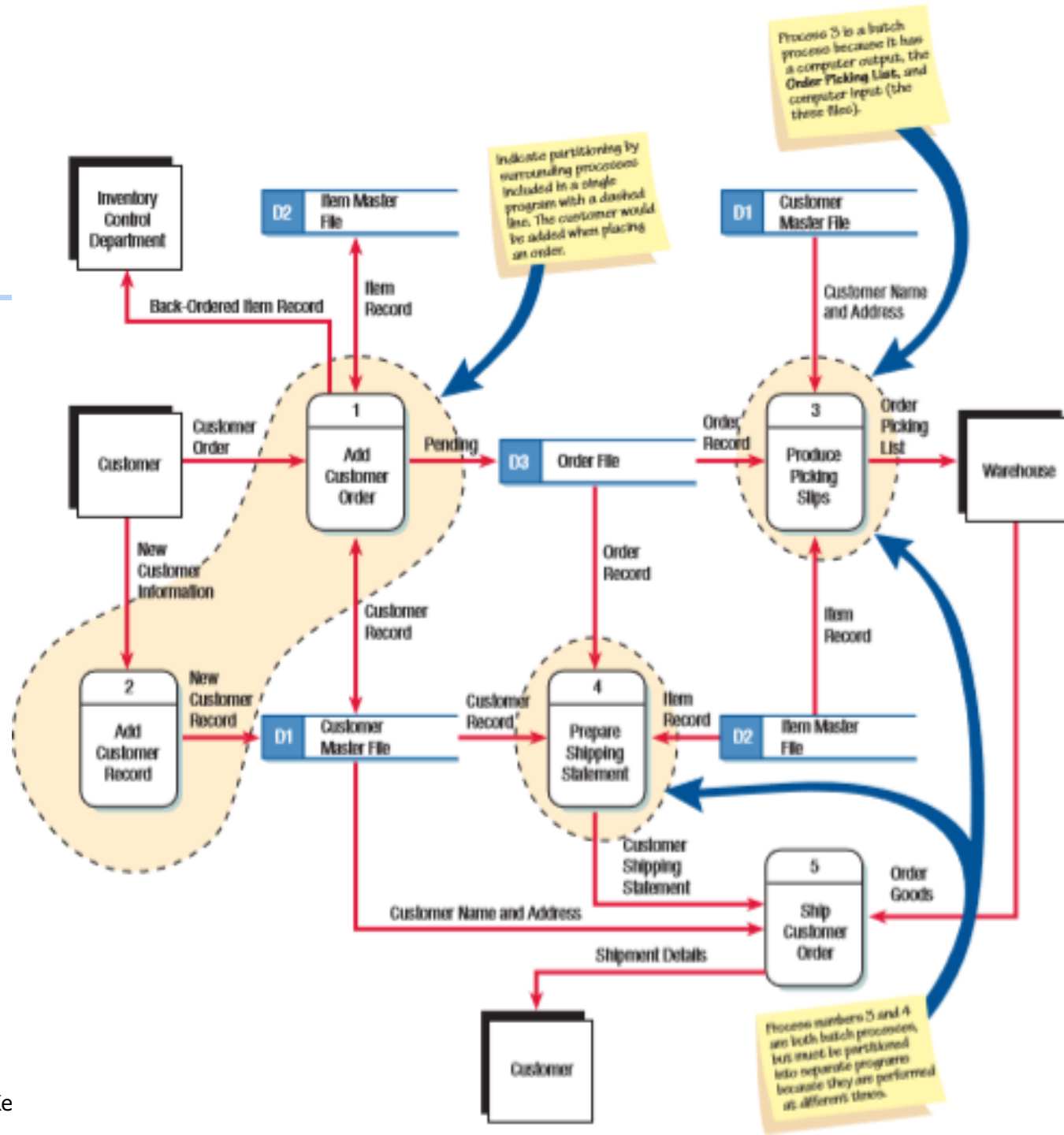
Partitioning Data Flow Diagrams

- Partitioning is the process of examining a data flow diagram and determining how it should be divided into collections of manual procedures and computer programs
- A dashed line is drawn around a process or group of processes that should be placed in a single computer program



Reasons for Partitioning

- Different user groups
- Timing
- Similar tasks
- Efficiency
- Consistency of data
- Security





Communicating Using Data Flow Diagrams

- Use unexploded data flow diagrams early when ascertaining information requirements
- Meaningful labels for all data components

Homework

- ❑ Produce physical **DFD** based on the logical DFD.

Summary

- Data flow diagrams
 - Structured analysis and design tools that allow the analyst to comprehend the system and subsystems visually as a set of interrelated data flows
- DFD symbols
 - Rounded rectangle
 - Double square
 - An arrow
 - Open-ended rectangle

Summary (continued)

- Creating the logical DFD
 - Context-level data flow diagram
 - Level 0 logical data flow diagram
 - Child diagrams
- Creating the physical DFD
 - Create from the logical data flow diagram
 - Partitioned to facilitate programming

Summary (continued)

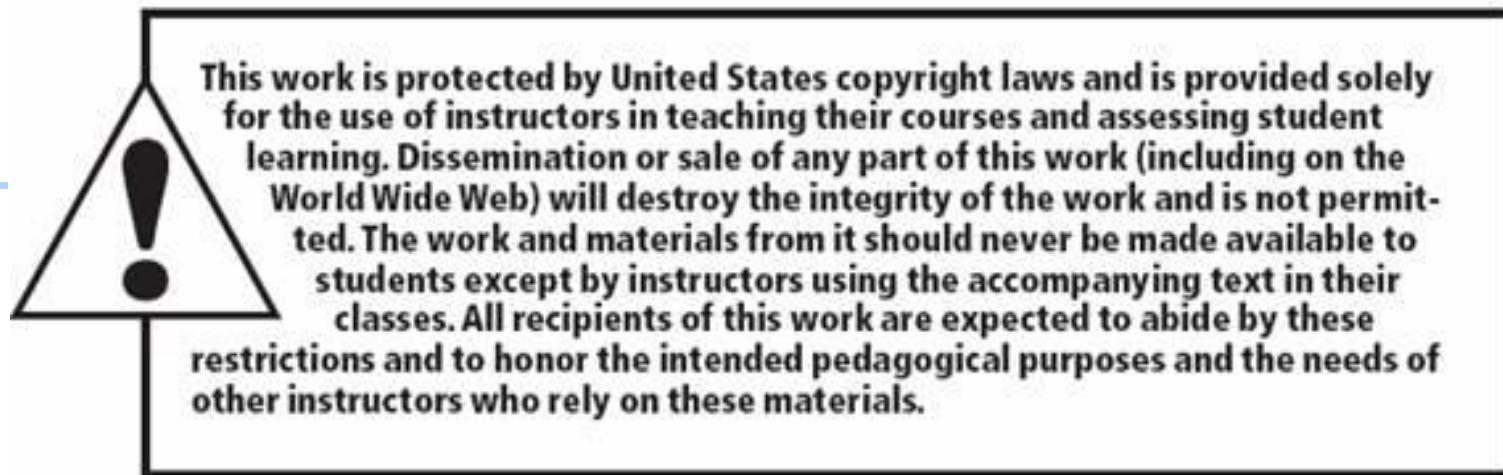
- Partitioning data flow diagrams
 - Whether processes are performed by different user groups
 - Processes execute at the same time
 - Processes perform similar tasks
 - Batch processes can be combined for efficiency of data
 - Processes may be partitioned into different programs for security reasons





Advantages of the Data Flow Approach

- Freedom from committing to the technical implementation too early
- Understanding of the interrelatedness of systems and subsystems
- Communicating current system knowledge to users
- Analysis of the proposed system



Copyright © 2014 Pearson Education, Inc.
Publishing as Prentice Hall