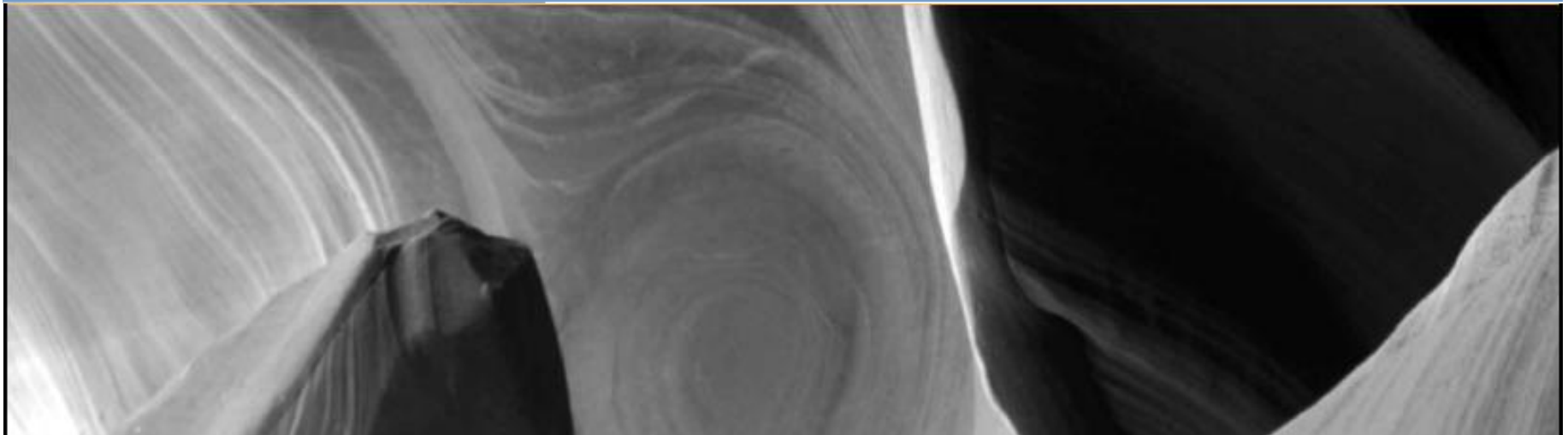


# *Systems Analysis and Design in a Changing World, Fifth Edition*

## CHAPTER

# 6

## THE TRADITIONAL APPROACH TO REQUIREMENTS



# Learning Objectives

- ◆ Explain how the traditional approach and the object-oriented approach differ when modeling the details of a use case
- ◆ List the components of a traditional system and the symbols representing them on a data flow diagram
- ◆ Describe how data flow diagrams can show the system at various levels of abstraction

# Learning Objectives (continued)

- ◆ Develop data flow diagrams, data element definitions, data store definitions, and process descriptions
- ◆ Develop tables to show the distribution of processing and data access across system locations

# Overview

- ◆ What the system does and what event occurs – activities and interactions (use case)
- ◆ Traditional structured approach to representing activities and interactions
- ◆ Diagrams and other models of the traditional approach
- ◆ RMO customer support system example shows how each model is related

# Traditional vs. Object-Oriented Approaches

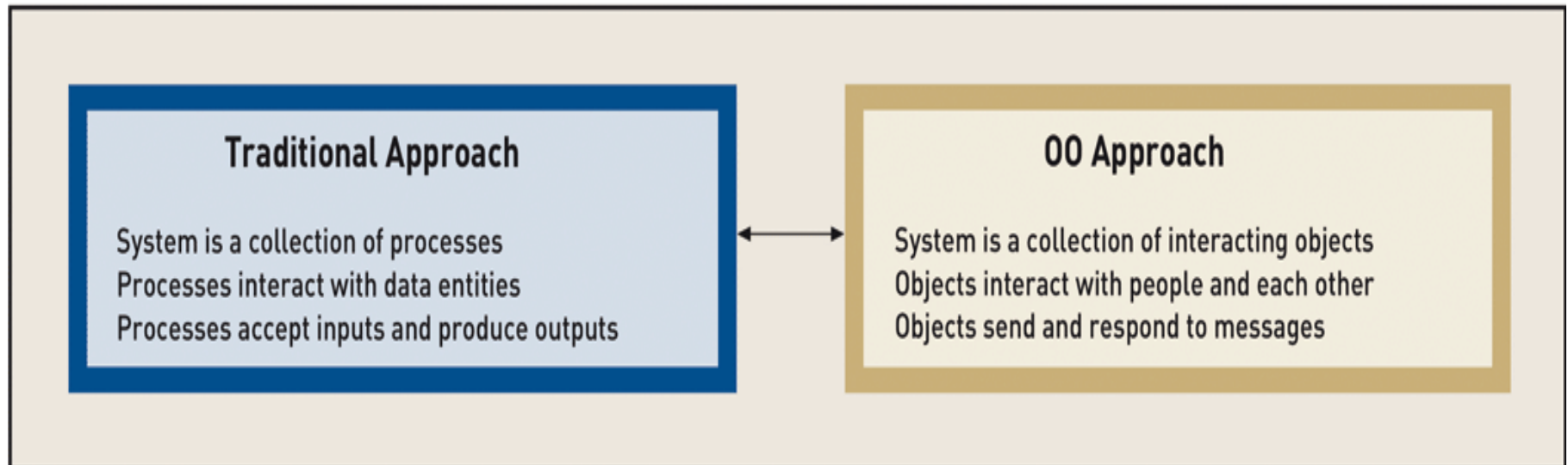


Figure 6-1

# Requirements for the Traditional and OO Approaches

**Figure 6-2**

Requirements models for the traditional and OO approaches

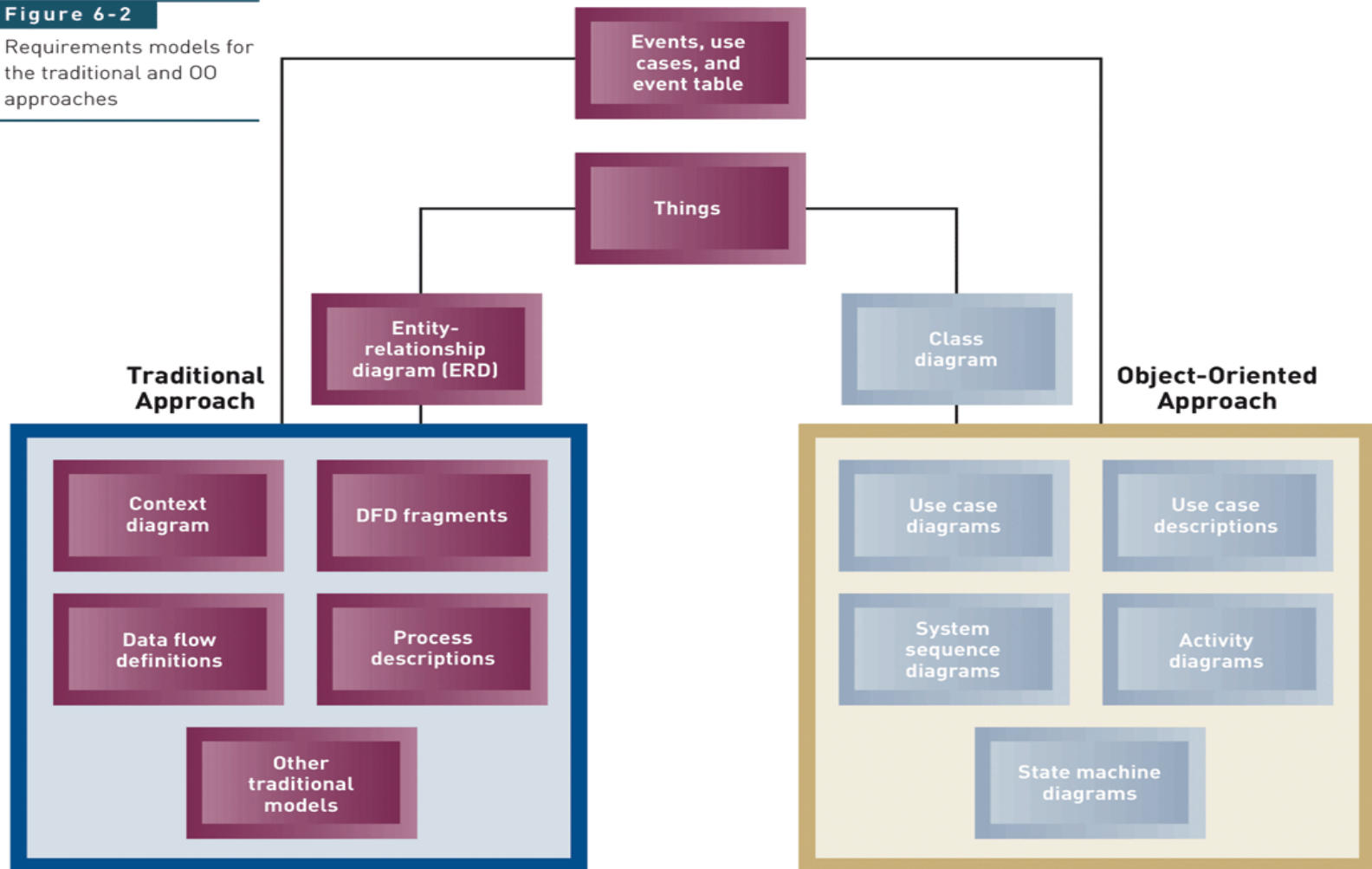


Figure 6-2

# Data Flow Diagrams (DFDs)

- ◆ Graphical system model that shows all main requirements for an IS in one diagram
  - Inputs/outputs
  - Processes
  - Data storage
- ◆ Easy to read and understand with minimal training

# Data Flow Diagram Symbols

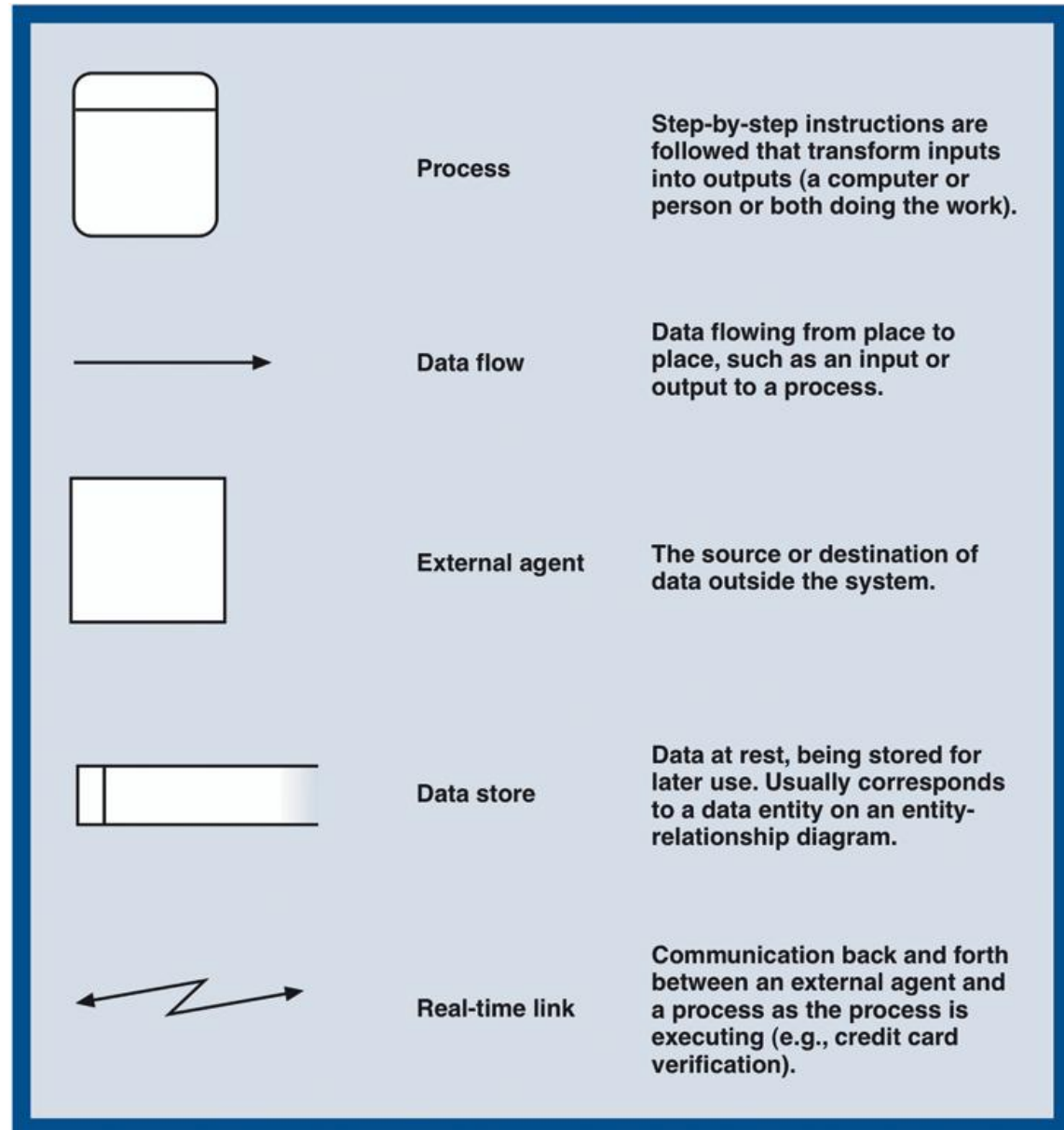


Figure 6-3



# DFD Fragment Showing Use Case *Look Up Item Availability* from the RMO

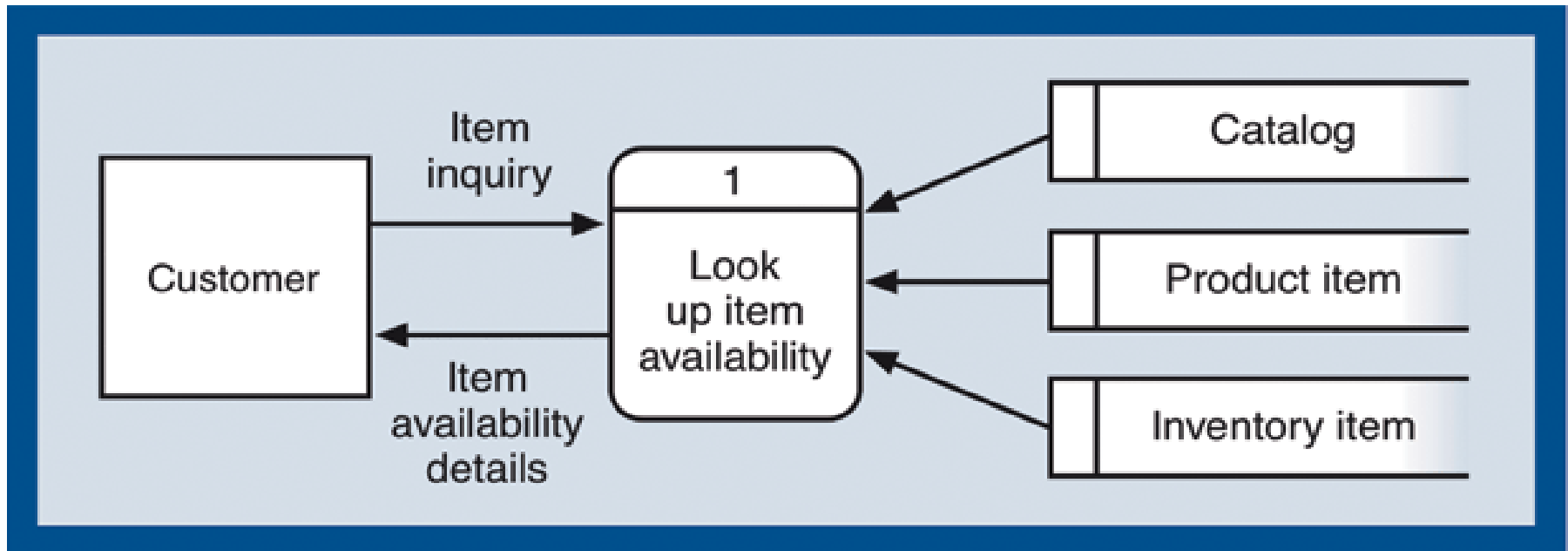


Figure 6-4

# DFD Integrates Event Table and ERD

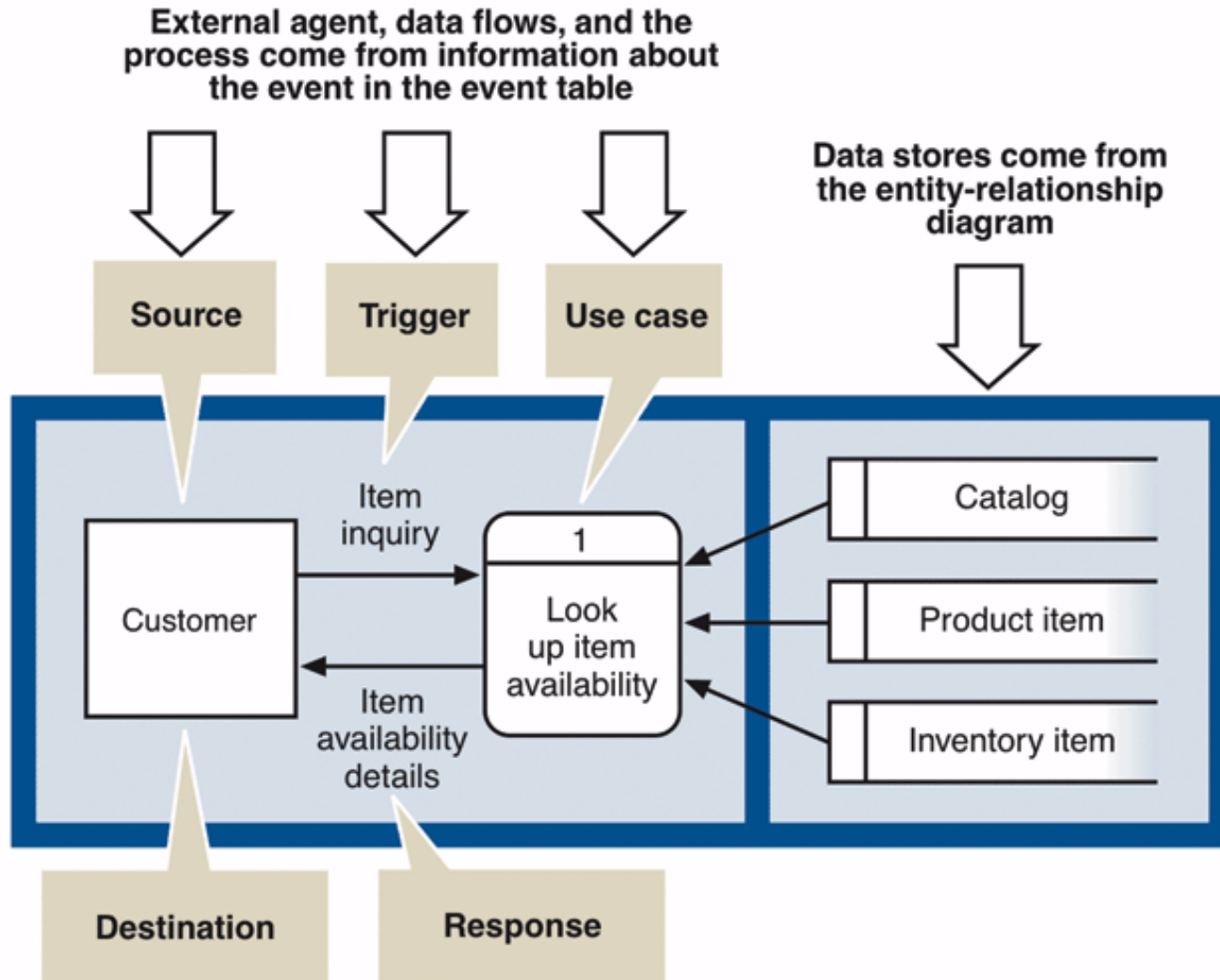


Figure 6-5

# DFD and Levels of Abstraction

- ◆ Data flow diagrams (DFDs) are decomposed into additional diagrams to provide multiple levels of detail
- ◆ Higher-level diagrams provide general views of system
- ◆ Lower-level diagrams provide detailed views of system
- ◆ Differing views are called levels of abstraction

# Layers of DFD

## Abstraction for Course Registration System

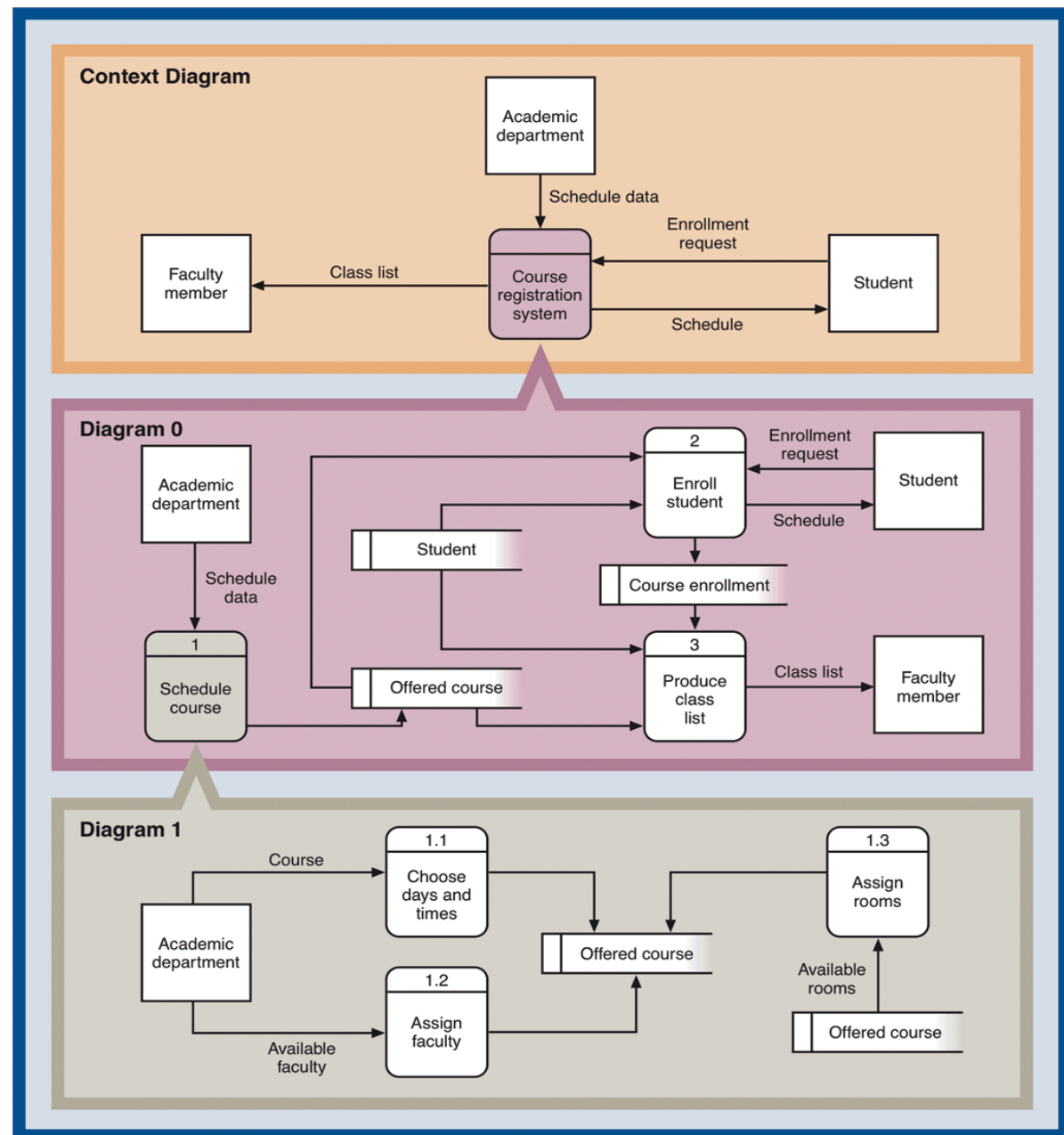


Figure 6-6

# Context Diagrams

- ◆ DFD that summarizes all processing activity for the system or subsystem
- ◆ Highest level (most abstract) view of system
- ◆ Shows system boundaries
- ◆ System scope is represented by a single process, external agents, and all data flows into and out of the system

# DFD Fragments

- ◆ Created for each use case in the event table
- ◆ Represent system response to one event within a single process symbol
- ◆ Self-contained models
- ◆ Focus attention on single part of system
- ◆ Show only data stores required in the use case

# Three Separate DFD Fragments for Course Registration System

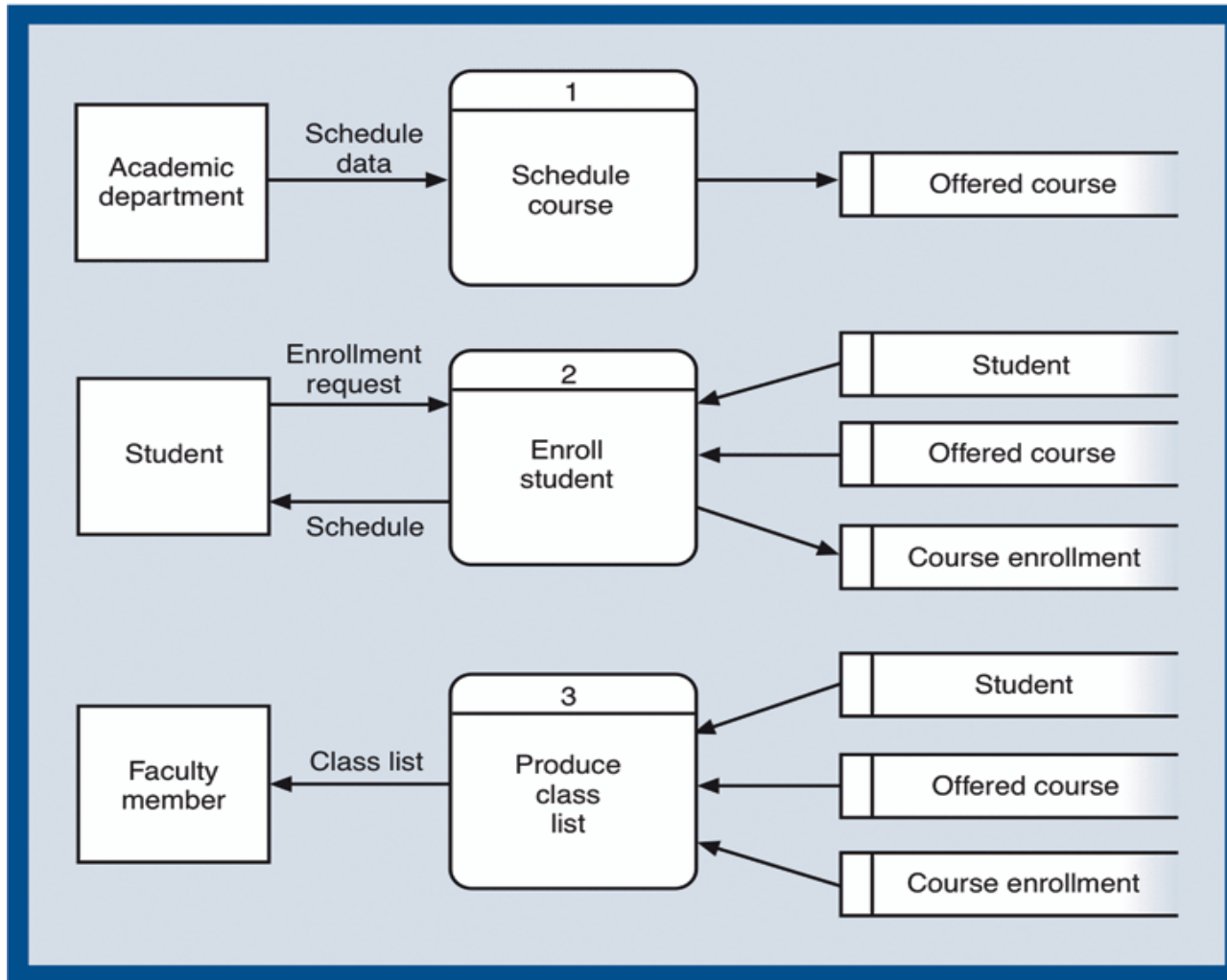


Figure 6-7

# Event-Partitioned System Model

- ◆ DFD to model system requirements using single process for each use case/activity in system or subsystem
- ◆ Combines all DFD fragments together to show decomposition of the context-level diagram
- ◆ Sometimes called “diagram 0”
- ◆ Used primarily as a presentation tool
- ◆ Decomposed into more detailed DFD fragments



# Combining DFD Fragments to Create Event-Partitioned System Model

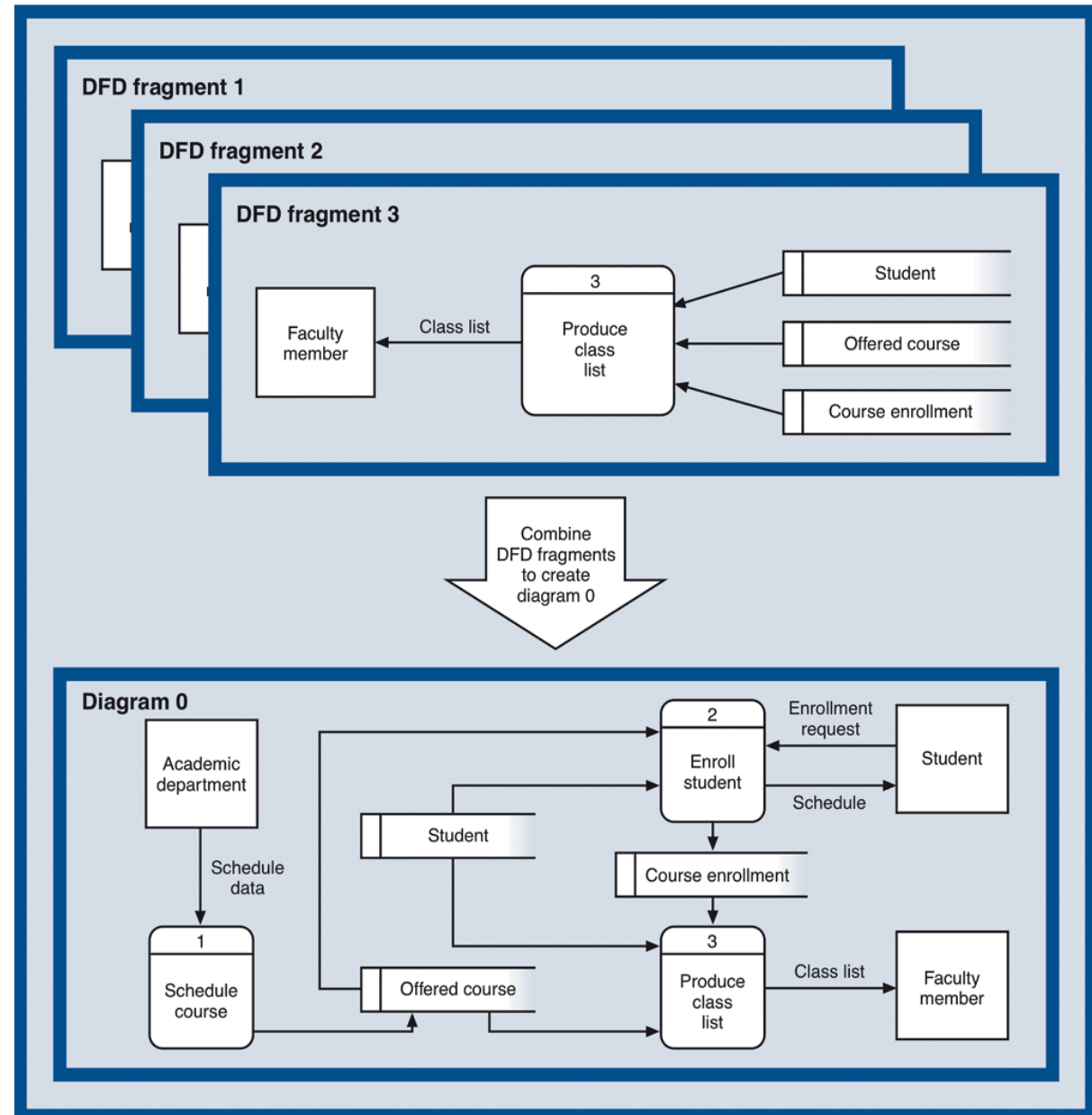


Figure 6-8

# Context Diagram for RMO Customer Support System

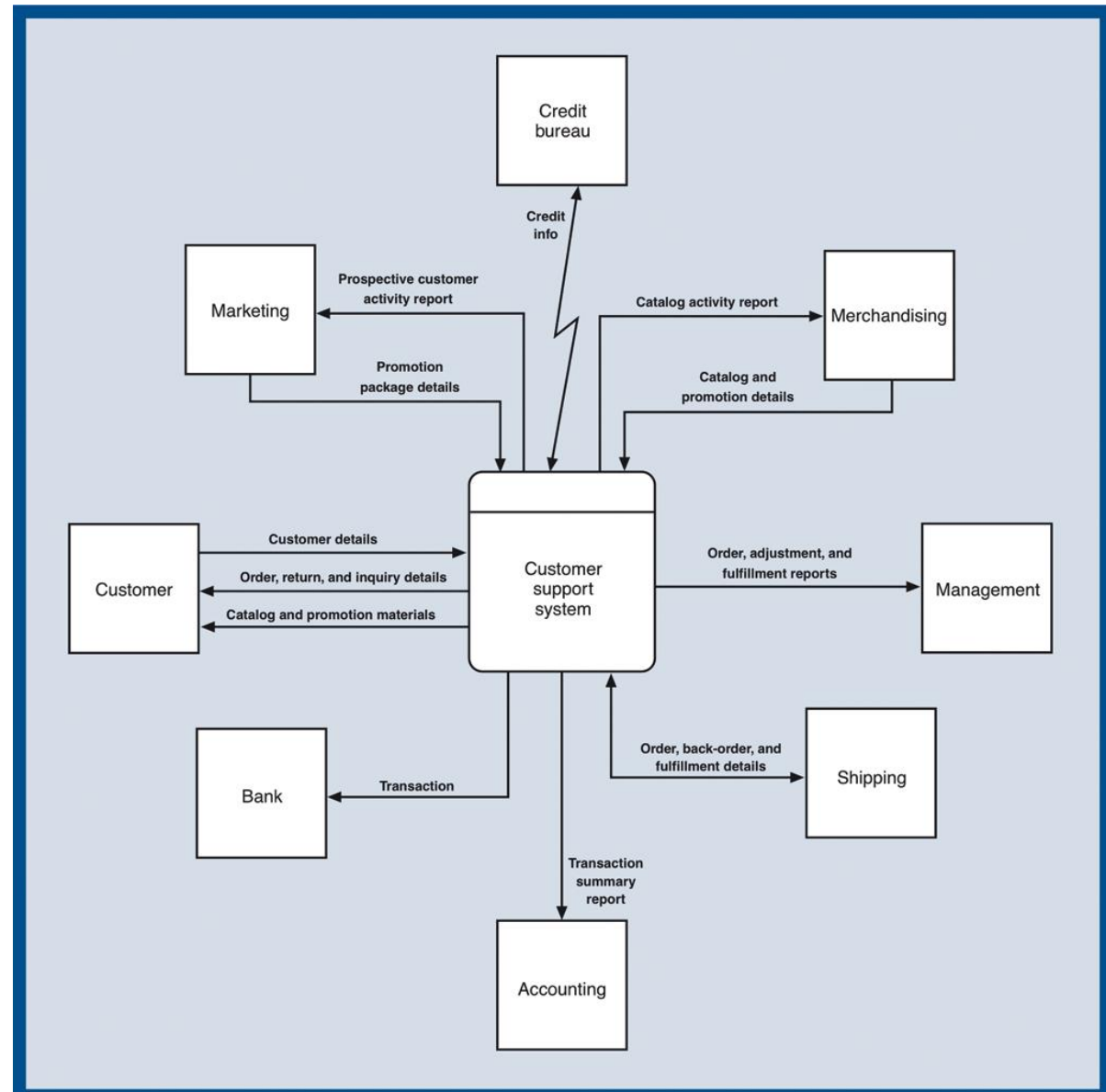


Figure 6-9

# RMO Subsystems and Use Cases/Activities from Event Table

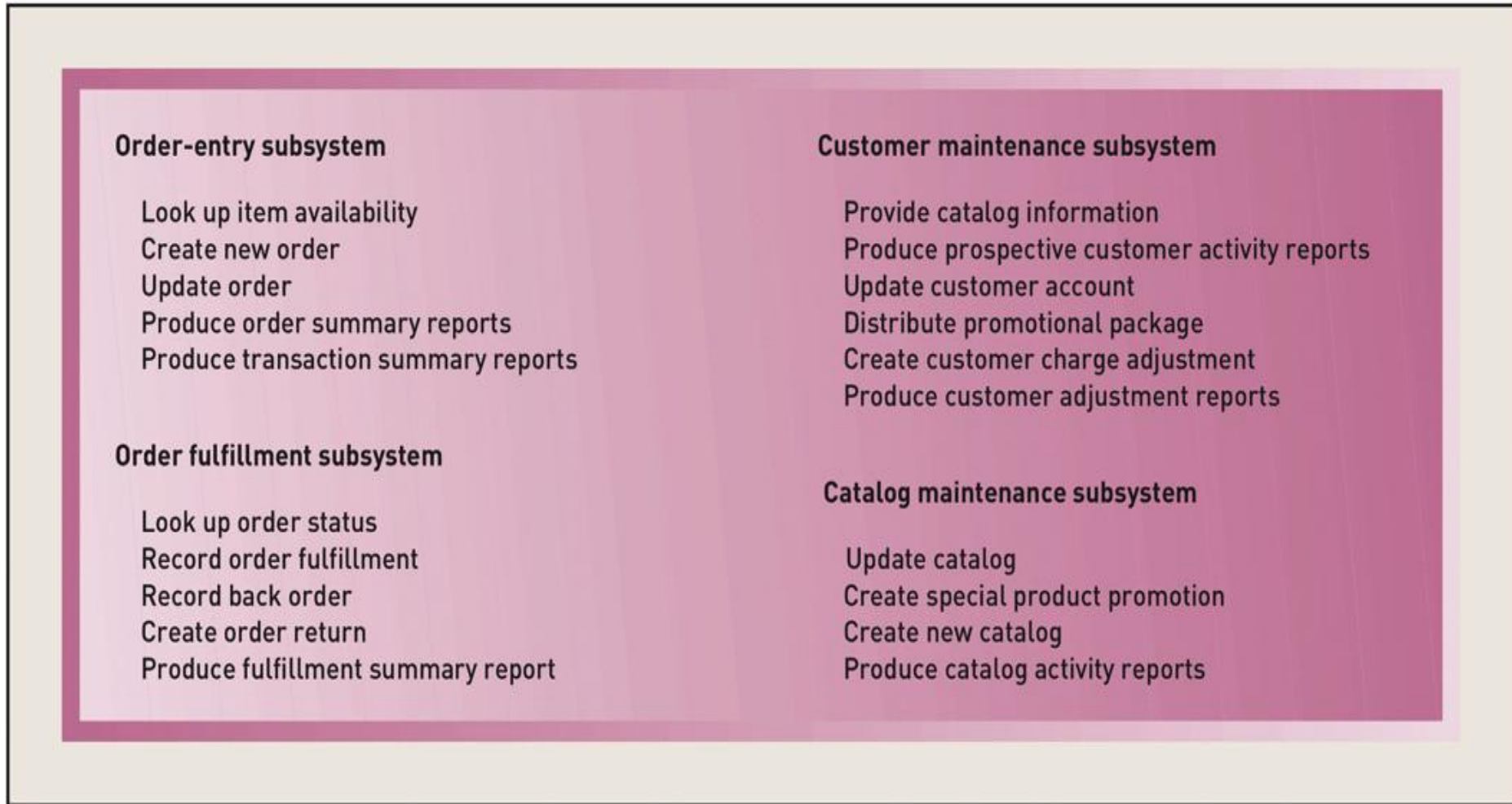


Figure 6-10

# Context Diagram for RMO Order-Entry Subsystem

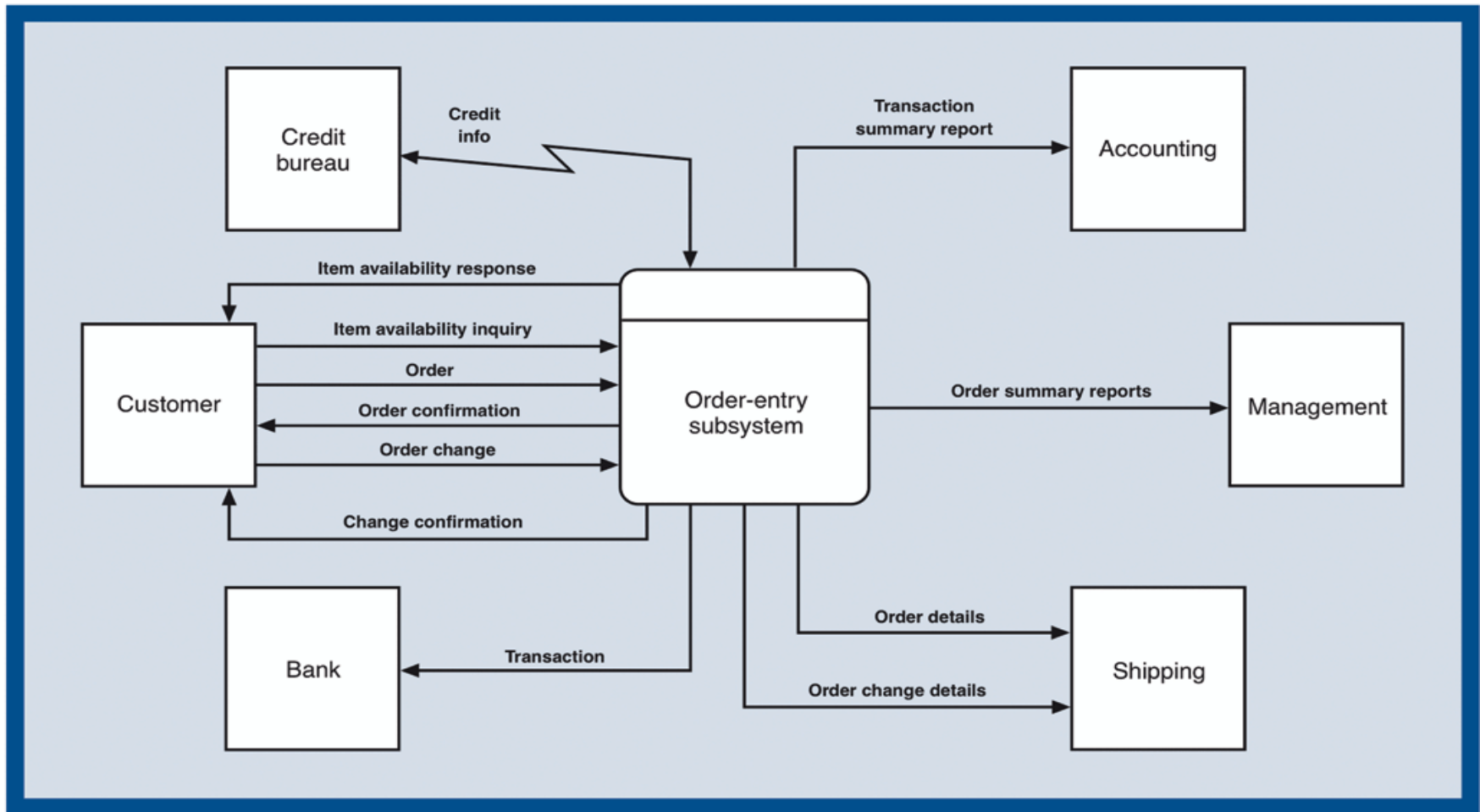


Figure 6-11

# Five Separate DFD Fragments for RMO

## Order-Entry Subsystem

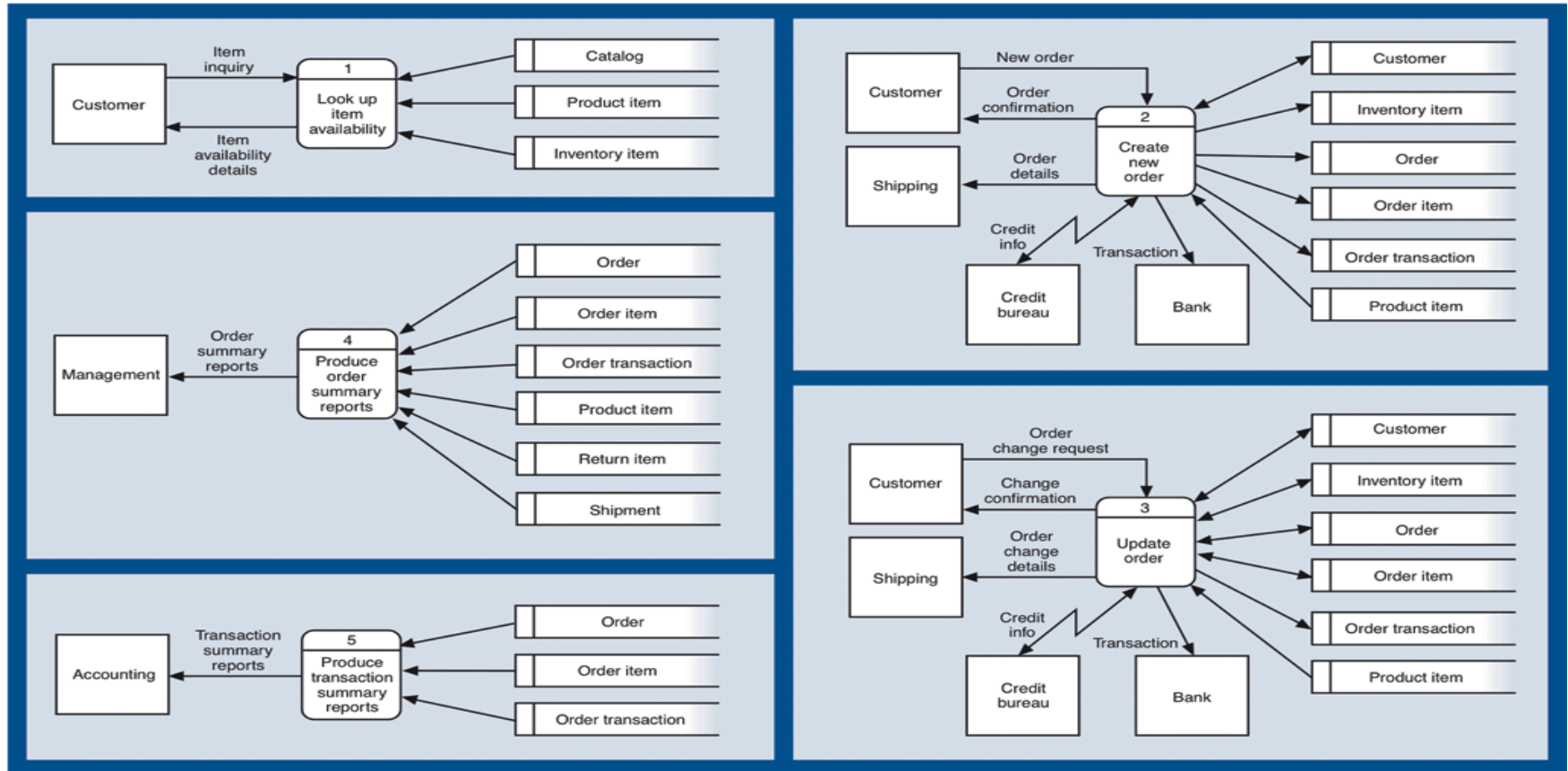


Figure 6-12

# Decomposing DFD Fragments

- ◆ Most DFD fragments can be further described using structured English
- ◆ Sometimes DFD fragments need to be diagrammed in more detail
- ◆ Decomposed into subprocesses in a detailed DFD
- ◆ DFD numbering scheme
  - Hierarchical decomposition
    - ◆ DFD Fragment 2 is decomposed into Diagram 2
    - ◆ Diagram 2 has processes 2.1, 2.2, 2.3, 2.4



# Detailed DFD for *Create new order* DFD Fragment

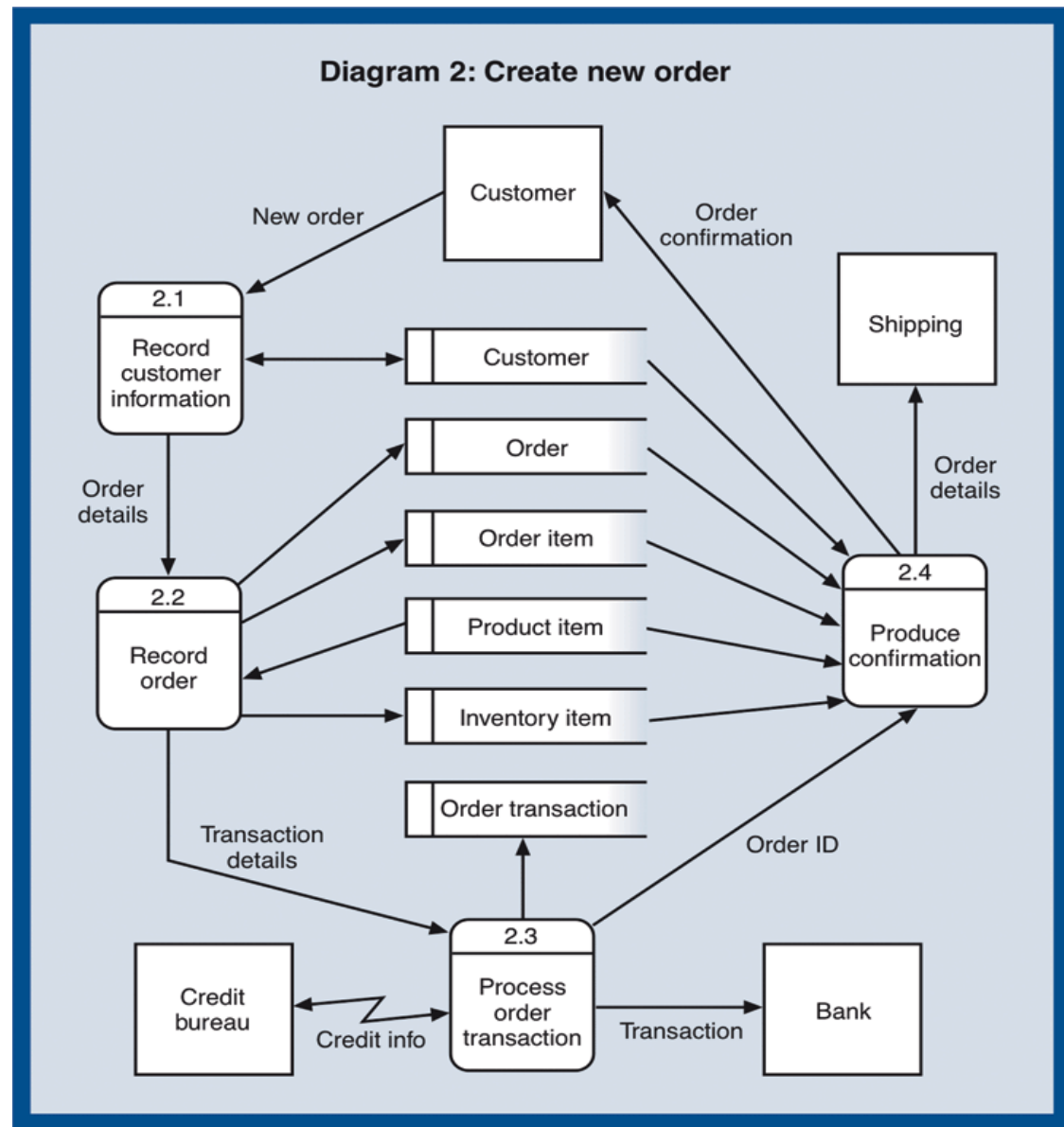


Figure 6-14

# Physical and Logical DFDs

## ◆ Logical model

- Assumes implementation in perfect technology
- Does not tell how system is implemented

## ◆ Physical model

- Describes assumptions about implementation technology
- Developed in last stages of analysis or in early design



# Physical DFD for Scheduling Courses

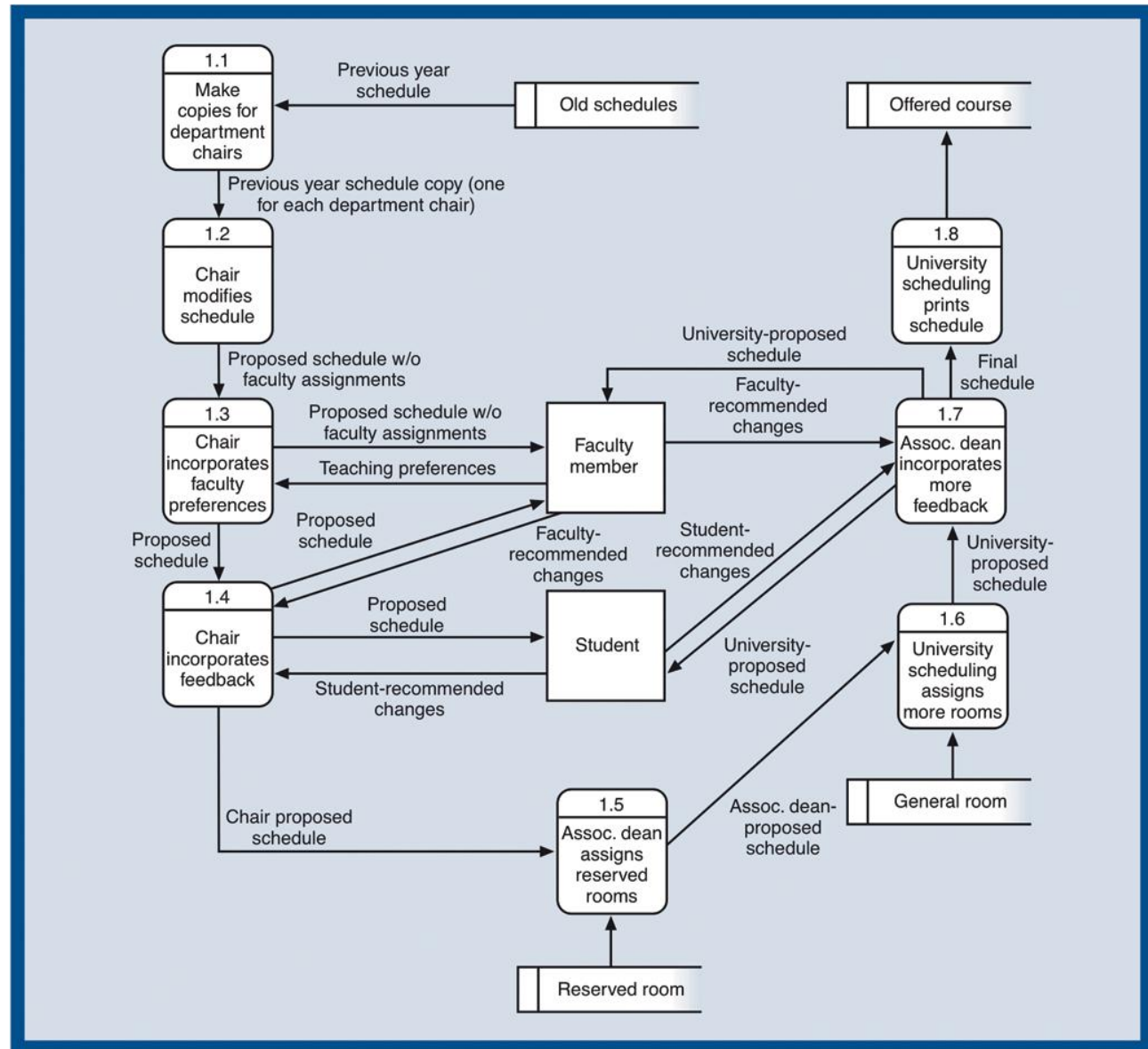


Figure 6-15

# Evaluating DFD Quality

- ◆ Readable
- ◆ Internally consistent and balanced
- ◆ Accurately represents system requirements
- ◆ Reduces information overload – rule of 7 +/- 2
  - Single DFD should not have more than 7 +/-2 processes
  - No more than 7 +/- 2 data flows should enter or leave a process or data store in a single DFD
- ◆ Minimizes required number of interfaces

# Data Flow Consistency Problems

- ◆ Differences in data flow content between a process and its process decomposition
- ◆ Data outflows without corresponding inflows
- ◆ Data inflows without corresponding outflows
- ◆ Results in unbalanced DFDs

# Consistency Rules

- ◆ All data that flows into a process must
  - Flow out of the process, or
  - Be used to generate data that flows out of the process
- ◆ All data that flows out of a process must
  - Have flowed into the process, or
  - Have been generated from data that flowed into the process

# Unnecessary Data Input: Black Hole

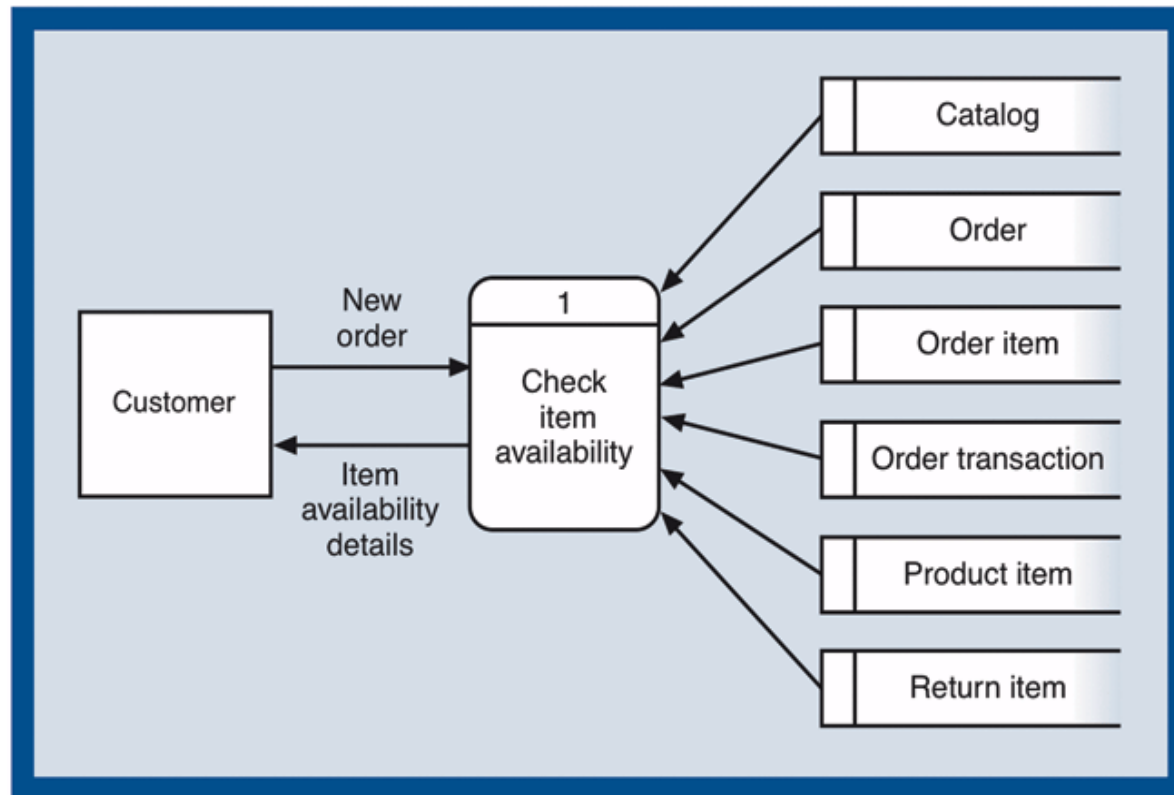


Figure 6-16

# Process with Impossible Data

## Output: a Miracle

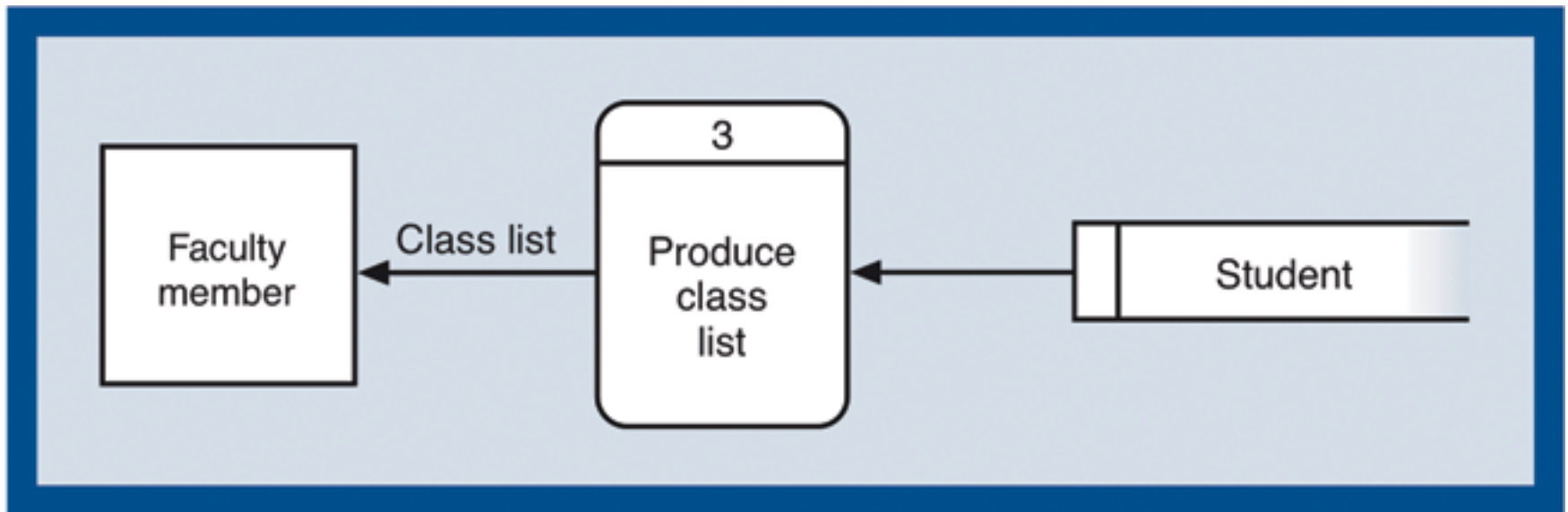


Figure 6-17

# Process with Unnecessary Data Input

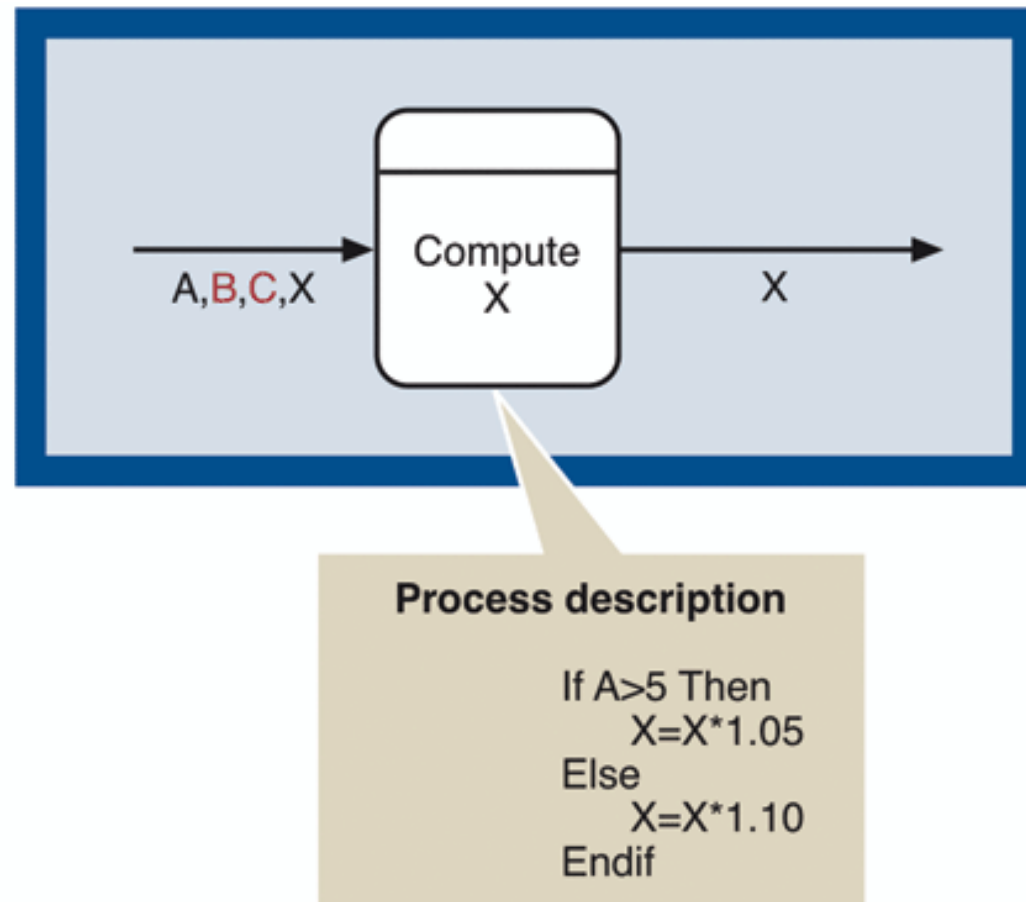


Figure 6-18

# Process with Impossible Data Output

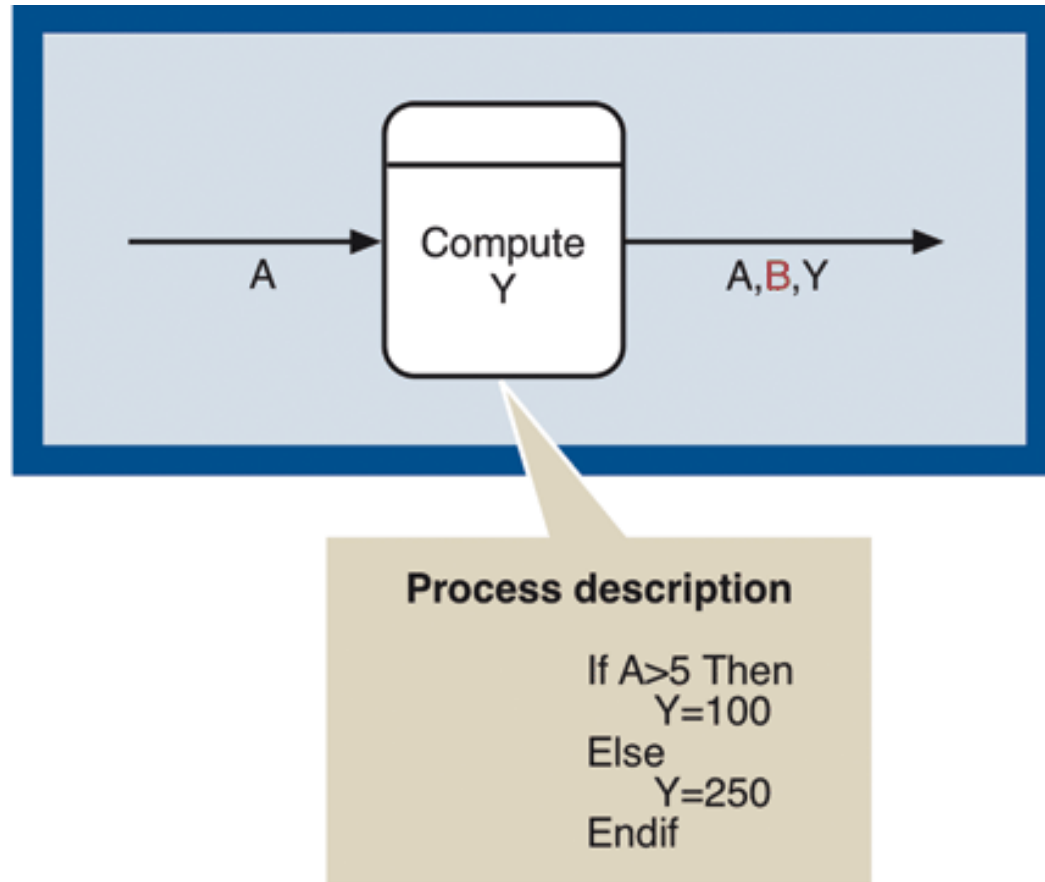


Figure 6-19



# Documentation of DFD Components

- ◆ Lowest-level processes need to be described in detail
- ◆ Data flow contents need to be described
- ◆ Data stores need to be described in terms of data elements
- ◆ Each data element needs to be described
- ◆ Various options for process definition exist

# Structured English

- ◆ Method of writing process specifications
- ◆ Combines structured programming techniques with narrative English
- ◆ Well-suited for lengthy sequential processes or simple control logic (single loop or if-then-else)
- ◆ Ill-suited for complex decision logic or few (or no) sequential processing steps

# Structured English Example

## Process Ballots Procedure

```
Collect all ballots
Place all ballots in a stack
Set Yes count and No count to zero
Repeat for each ballot in the stack
    If Yes is checked then
        Add one to Yes count
    Else
        Add one to No count
    Endif
    Place ballot on counted ballot stack
Endrepeat
If Yes count is greater than No count then
    Declare Yes the winner
Else
    Declare No the winner
Endif
Store the counted ballot stack in a safe place
End Process Ballots Procedure
```

Figure 6-20

# Process 2.1 and Structured English Process Description

## Process 2.1 - Record Customer Information

```

Ask if customer has an account (or has made a previous order)
If customer has an account then
    Ask for identification information
    Query database with identifying information
    Copy query response data to Order details
Else
    Create an empty Customer record in the database
    Ask customer for Customer attributes
    Update empty Customer record with Customer attributes
Endif
Ask customer for order information for first item
While more order items Do
    Update Order details with order information
Endwhile
  
```

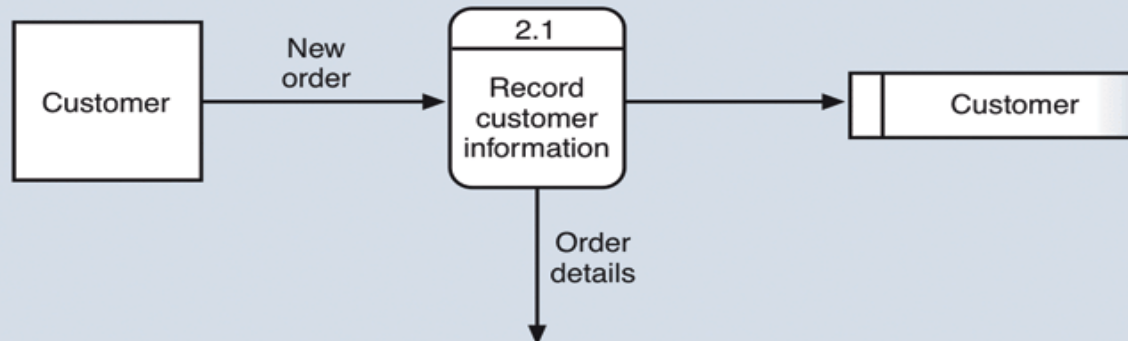


Figure 6-21

# Decision Tables and Decision Trees

- ◆ Can summarize complex decision logic better than structured English
- ◆ Incorporate logic into the table or tree structure to make descriptions more readable

# Decision Table for Calculating Shipping Charges

YTD purchases > \$250	YES						NO					
Number of Items (N)	N ≤ 3			N ≥ 4			N ≤ 3			N ≥ 4		
Delivery Day	Next	2nd	7th	Next	2nd	7th	Next	2nd	7th	Next	2nd	7th
Shipping Charge (\$)	25	10	N*1.50	N*6.00	N*2.50	Free	35	15	10	N*7.50	N*3.50	N*2.50

Figure 6-23

# Decision Tree for Calculating Shipping Charges

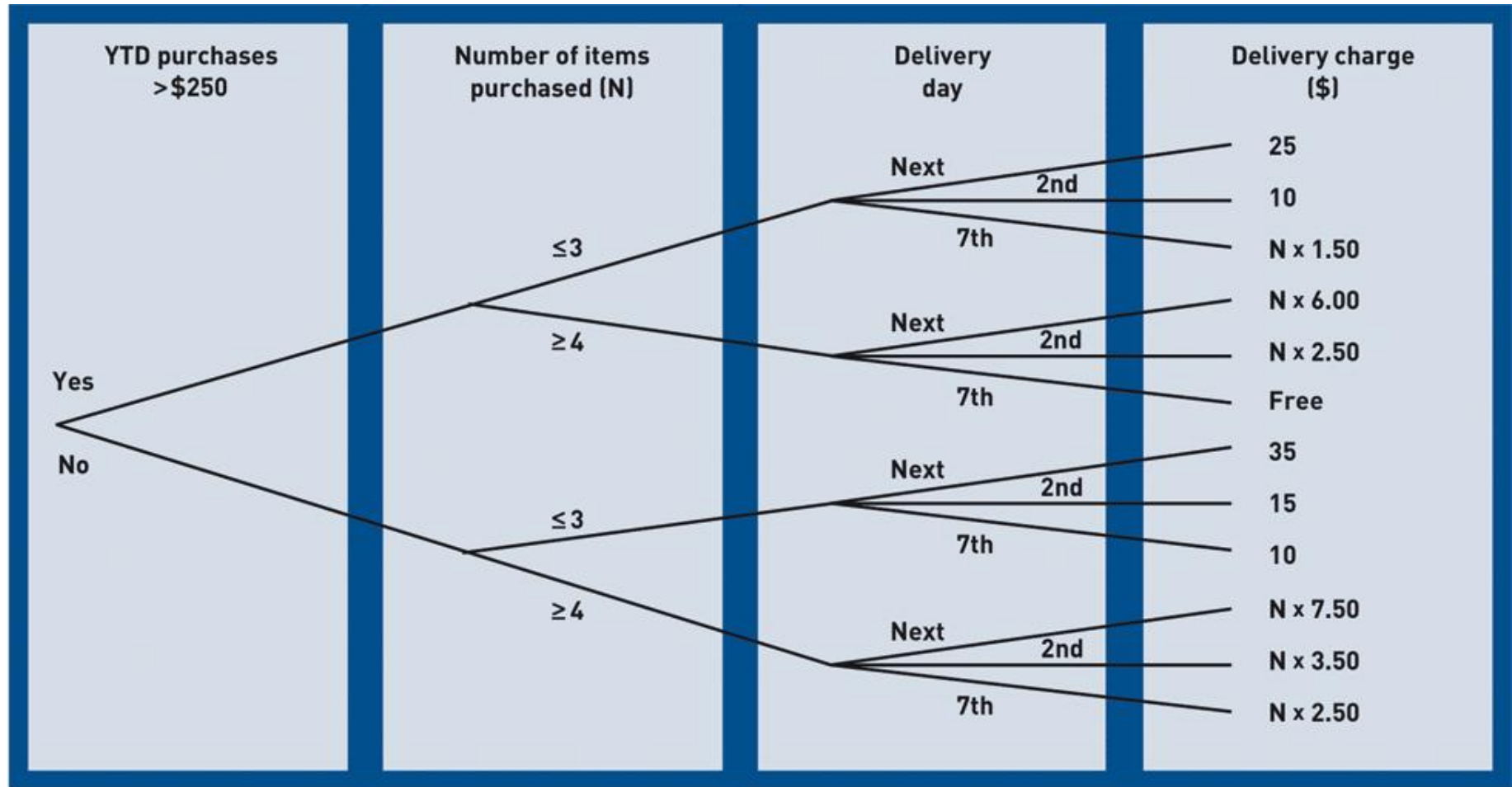


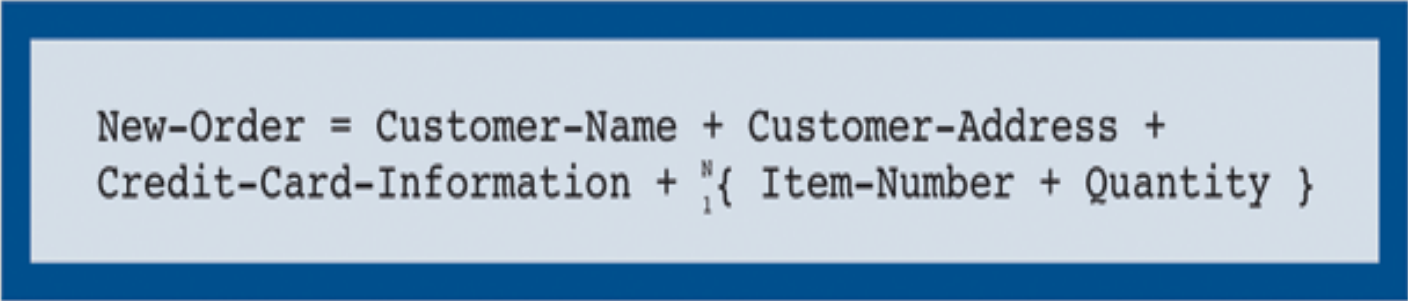
Figure 6-24

# Data Flow Definitions

- ◆ Textual description of data flow's content and internal structure
- ◆ Often coincide with attributes of data entities included in ERD plus computed values
- ◆ Algebraic notion describes data elements on data flow plus data structure



# Algebraic Notation for Data Flow Definition



```
New-Order = Customer-Name + Customer-Address +  
Credit-Card-Information +  $\sum_1$ { Item-Number + Quantity }
```

Figure 6-27

# Data Flow Definition for RMO Products and Items Control Break Report

```
products-and-items-report =  
  
  {  
    {  
      product-id + product-name + season + category +  
      supplier + unit-price + special + special-price +  
      discontinued + description +  
      {  
        size + color + style + units-in-stock +  
        recorder-level + units-on-order  
      }  
    }  
  }
```

Figure 6-29

# Data Element Definitions

- ◆ Data type description
  - String, integer, floating point, Boolean
  - Sometimes very specific written description
- ◆ Length of element
- ◆ Maximum and minimum values
- ◆ Data dictionary – repository for definitions of data flows, data stores, and data elements

# Data Element Definition Examples

```
units-in-stock =  
    a positive integer  
  
supplier =  
    a four digit numeric code  
  
unit-price =  
    a positive real number accurate to two decimal places,  
    always in U.S. dollars  
  
description =  
    a text field containing a maximum of 50 printable characters  
  
special =  
    a coded field with one of the following values  
    0: item is not "on special"  
    1: item is "on special"
```

Figure 6-30

# Components of a Traditional Analysis Model



Figure 6-31

# Locations and Communication Through Networks

- ◆ Logical information needed during analysis
  - Number of user locations
  - Processing and data access requirements at various locations
  - Volume and timing of processing and data access requests
- ◆ Needed to make initial design decisions such as
  - Distribution of computer systems, application software, database components, network capacity

# Gathering Location Information

- ◆ Identify locations where work is to be performed
- ◆ Draw location diagram
- ◆ List functions performed by users at each location
- ◆ Build activity-location matrix
  - Rows are system activities from event table
  - Columns are physical locations
- ◆ Build activity-data (CRUD) matrix
  - CRUD – create, read, update, and delete

# RMO Activity-Location Matrix

ACTIVITY	LOCATION				
	Corporate offices [Park City]	Distribution warehouses [Salt Lake City, Albuquerque, Portland]	Mail-order [Provo]	Phone sales [Salt Lake City]	Customer direct interaction [Anticipated]
Look up item availability	X	X	X	X	X
Create new order			X	X	X
Update order				X	X
Look up order status	X	X		X	X
Record order fulfillment		X			
Record back order		X			
Create order return		X			
Provide catalog info			X	X	X
Update customer account	X		X	X	X
Distribute promotional package	X				
Create customer charge adjustment	X				
Update catalog	X				
Create special product promotion	X				
Create new catalog	X				

Figure 6-33



# RMO Activity-Data Matrix (CRUD)

ACTIVITIES	DATA ENTITIES										
	Catalog	Customer	Inventory item	Order	Order item	Order transaction	Package	Product item	Return item	Shipment	Shipper
Look up item availability			R								
Create new order		CRU	RU	C	C	C	R	R		C	R
Update order		RU	RU	RUD	RUD	RUD	R	R		CRUD	R
Look up order status		R		R	R	R				R	R
Record order fulfillment					RU					RU	
Record back order					RU					CRU	
Create order return		CRU		RU		C			C		
Provide catalog info	R		R				R	R			
Update customer account		CRUD									
Distribute promotional package	R	R	R				R	R			
Create customer charge adjustment		RU				CRUD					
Update catalog	RU		R				RU	R			
Create special product promotion	R		R				R	R			
Create new catalog	C		R				CRU	R			

C = Creates new data, R = Reads existing data, U = Updates existing data, D = Deletes existing data

Figure 6-34

# Summary

- ◆ Data flow diagrams (DFDs) are used in combination with event table and entity-relationship diagram (ERD) to model system requirements
- ◆ DFDs model system as set of processes, data flows, external agents, and data stores
- ◆ DFDs easy to read – graphically represent key features of system using small set of symbols
- ◆ Many types of DFDs – context diagrams, DFD fragments, subsystem DFDs, event-partitioned DFDs, and detailed process DFDs

# Summary (continued)

- ◆ Each process, data flow, and data store requires detailed definition
- ◆ Analyst may define processes as structured English process specifications, decision tables, decision trees, or detail process DFDs
- ◆ Detailed process decomposition DFDs used when internal process complexity is great
- ◆ Data flows are defined by component data elements and their internal structure (algebraic notation)