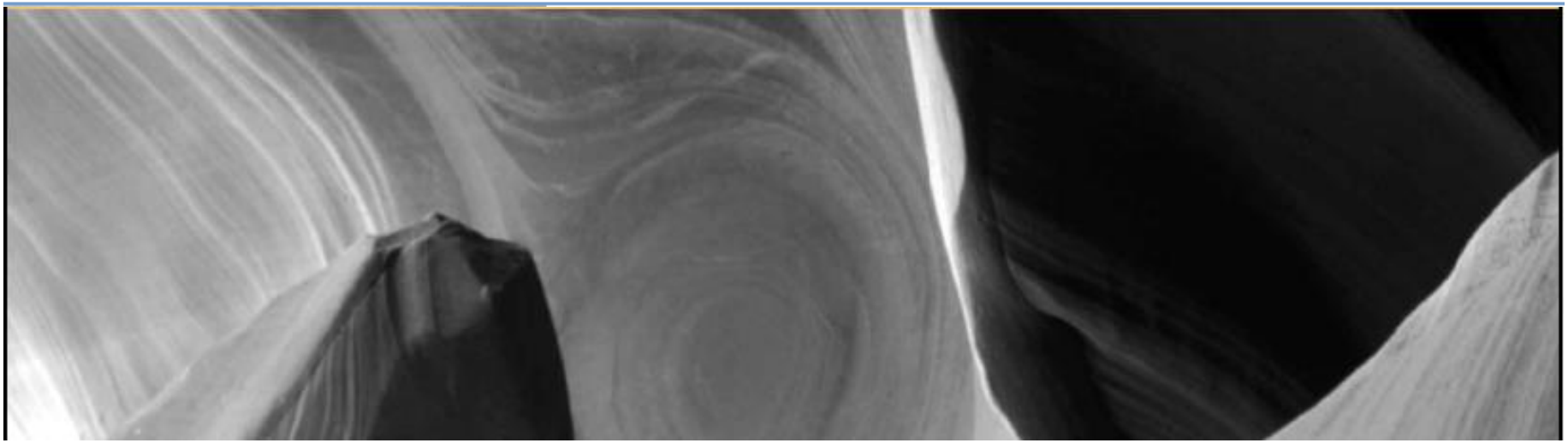
Systems Analysis and Design in a Changing World, Fifth Edition

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

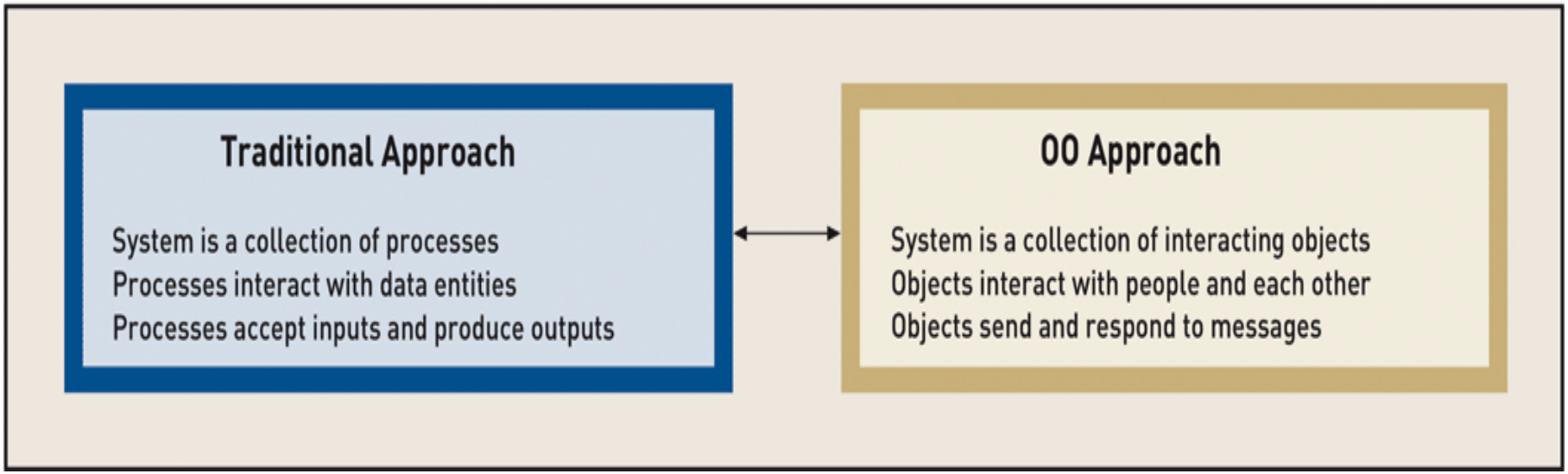


Figure 6-1

Requirements for the Traditional and OO Approaches

Figure 6-2
Requirements models for the traditional and OO approaches

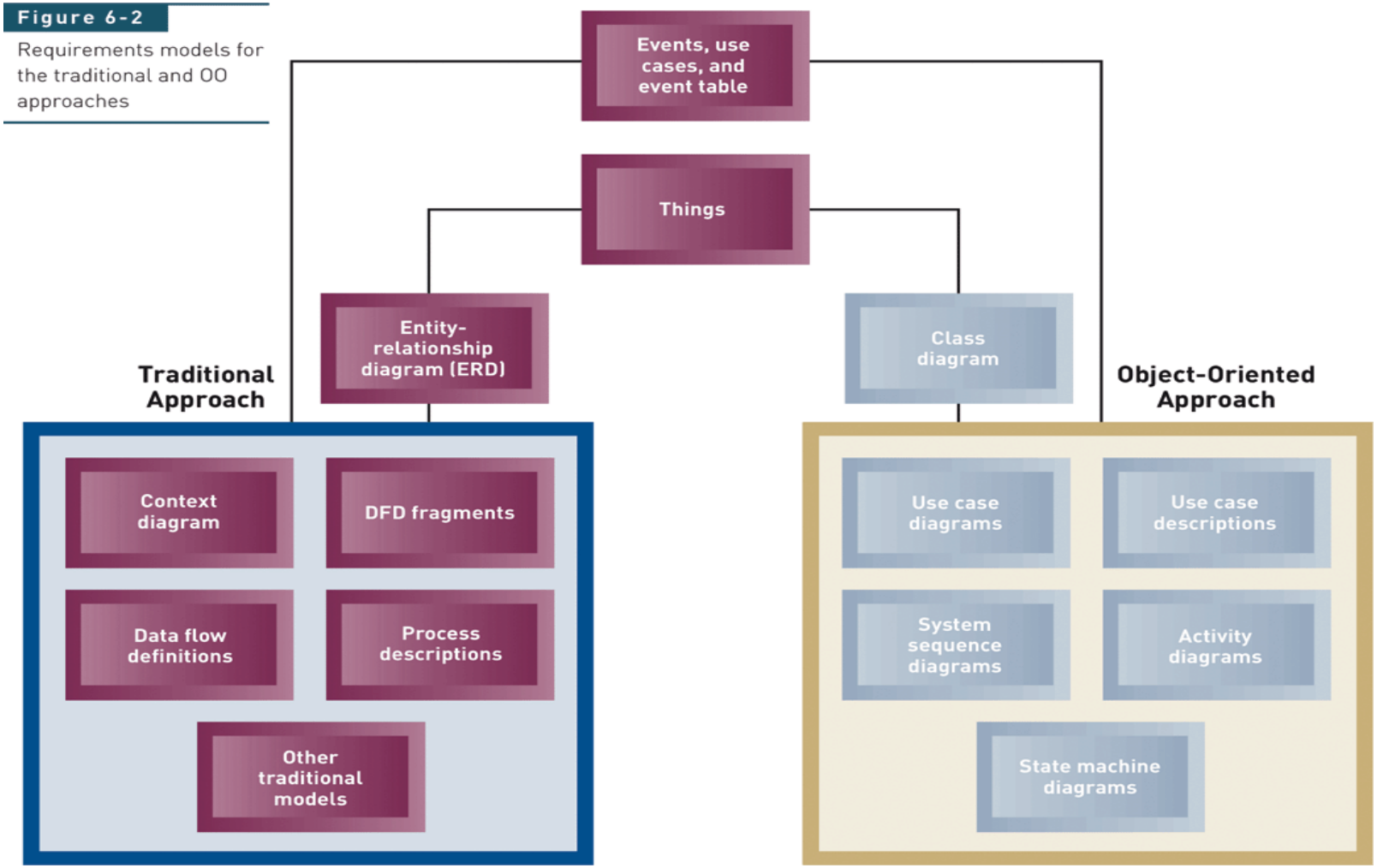


Figure 6-2

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

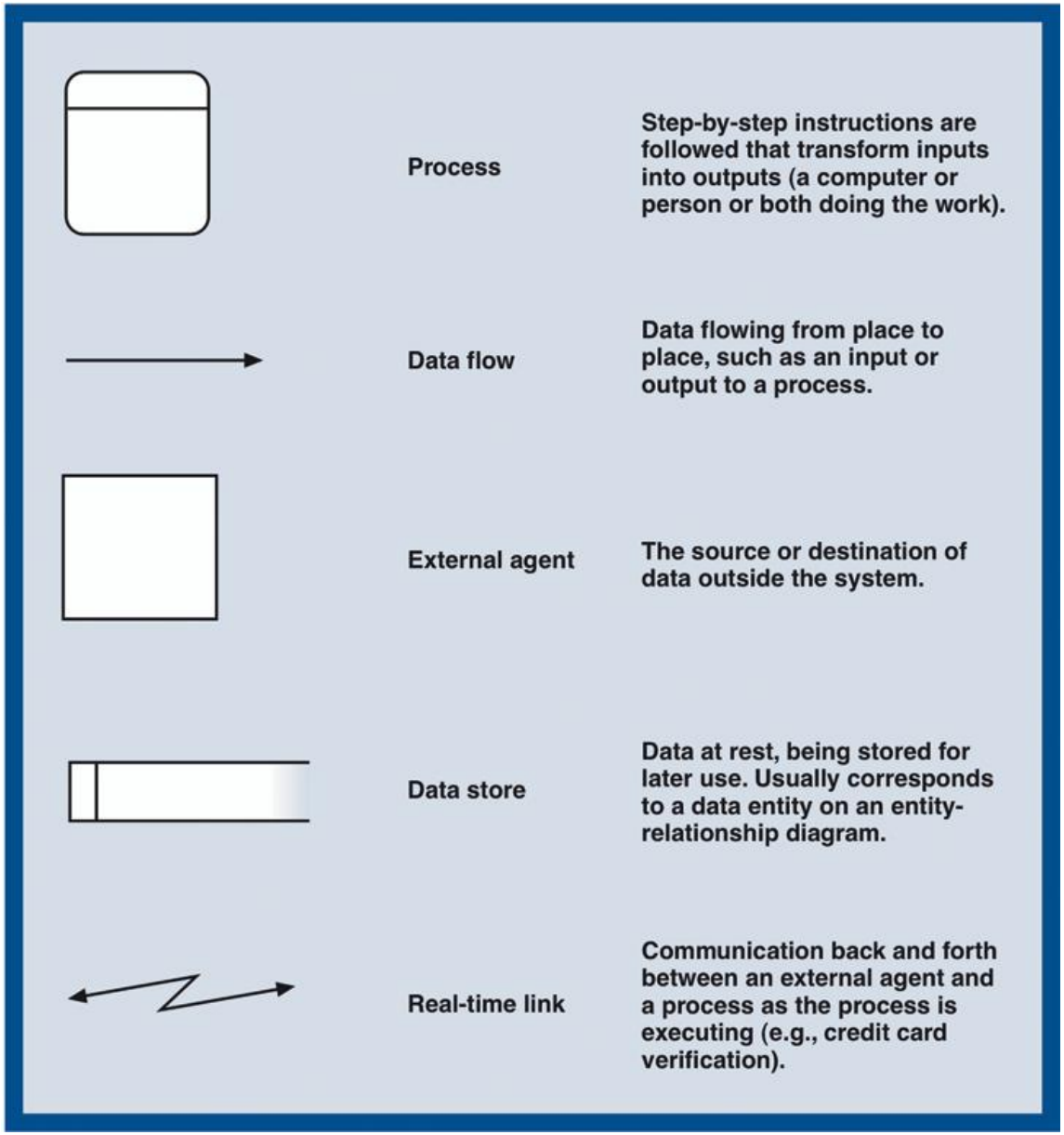


Figure 6-3

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

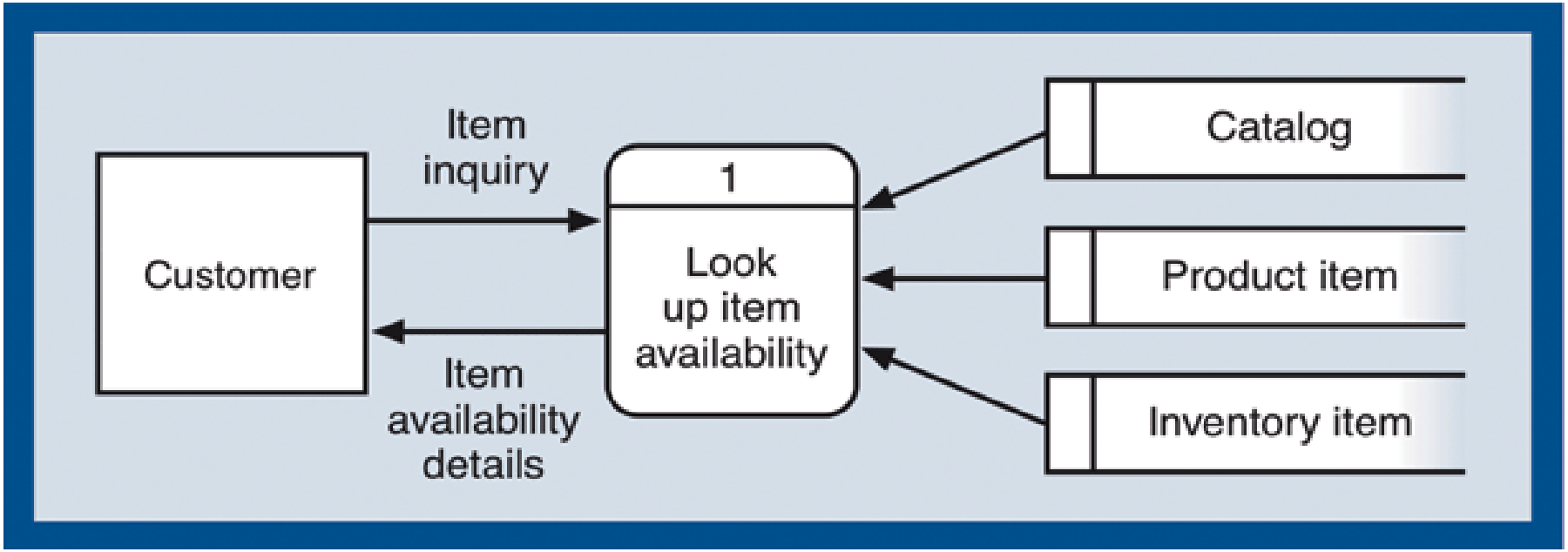


Figure 6-4

DFD Integrates Event Table and ERD

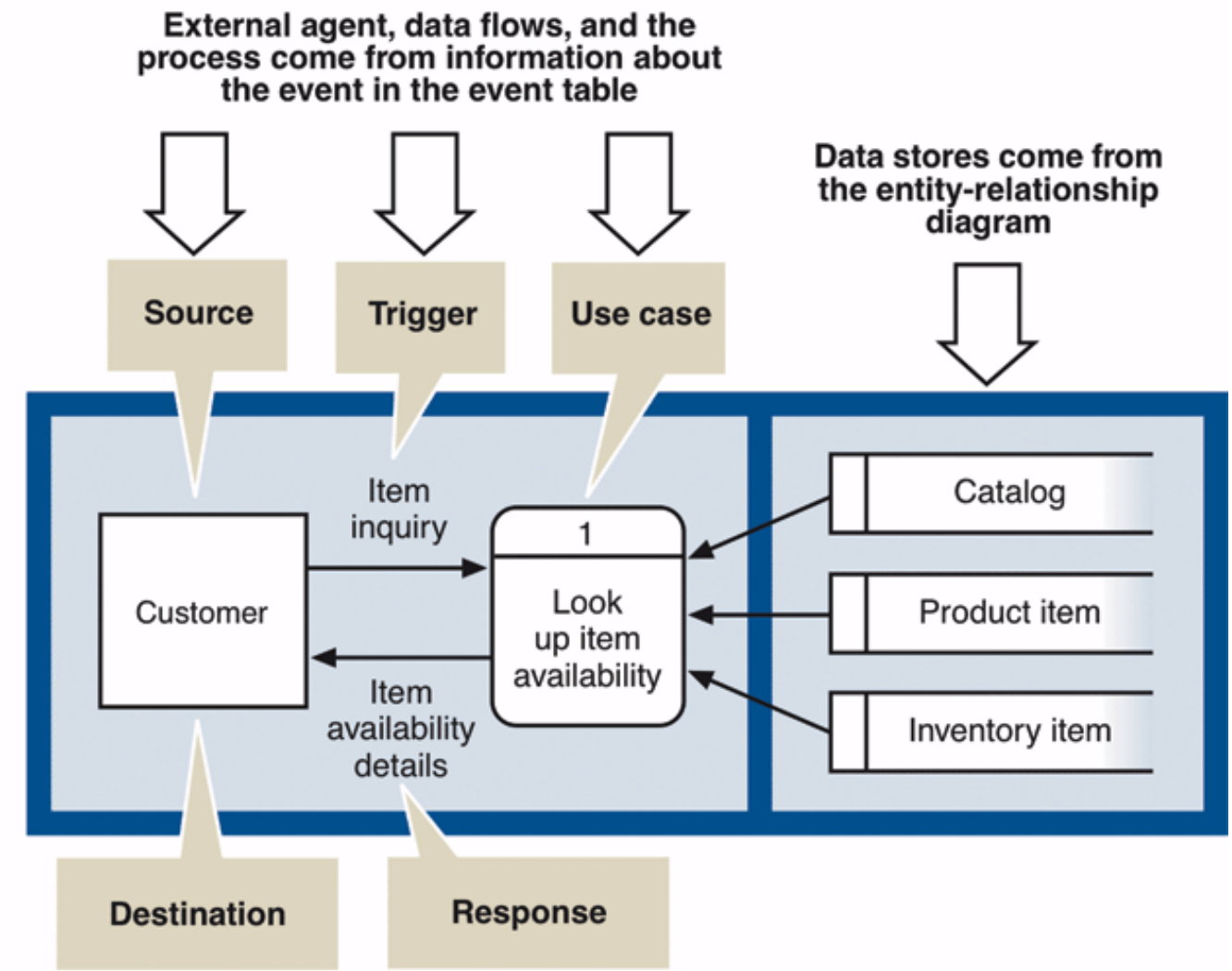


Figure 6-5

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

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

Context Diagram for RMO Customer Support System

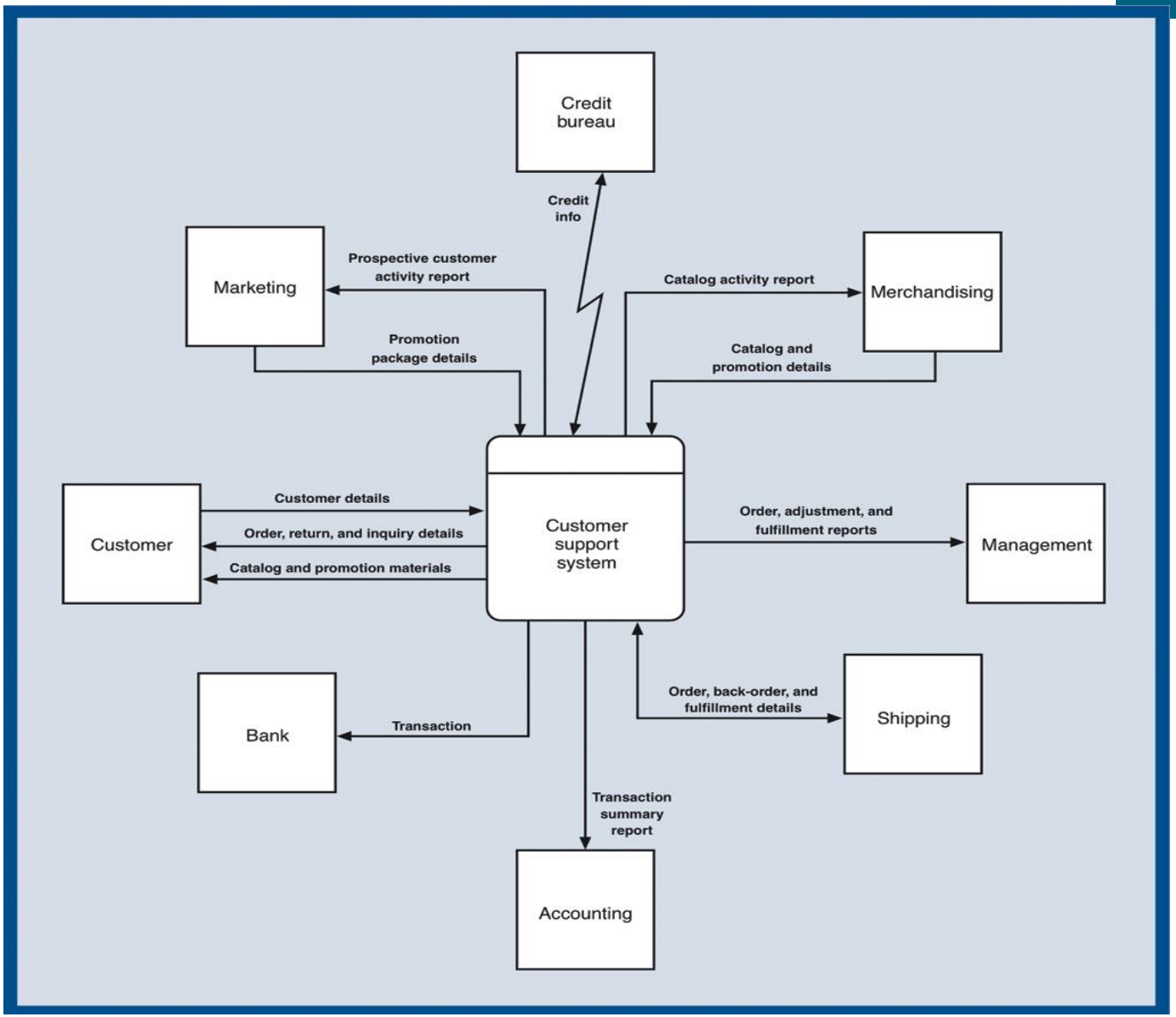


Figure 6-9

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

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 the DFD diagram (overview-level)
- Sometimes called “diagram 0”
- Used primarily as a presentation tool
- Decomposed into more detailed DFD fragments

RMO Subsystems and Use Cases/Activities from Event Table

Order-entry subsystem

- Look up item availability
- Create new order
- Update order
- Produce order summary reports
- Produce transaction summary reports

Order fulfillment subsystem

- Look up order status
- Record order fulfillment
- Record back order
- Create order return
- Produce fulfillment summary report

Customer maintenance subsystem

- Provide catalog information
- Produce prospective customer activity reports
- Update customer account
- Distribute promotional package
- Create customer charge adjustment
- Produce customer adjustment reports

Catalog maintenance subsystem

- Update catalog
- Create special product promotion
- Create new catalog
- Produce catalog activity reports

Figure 6-10

Context Diagram for RMO Order-Entry Subsystem

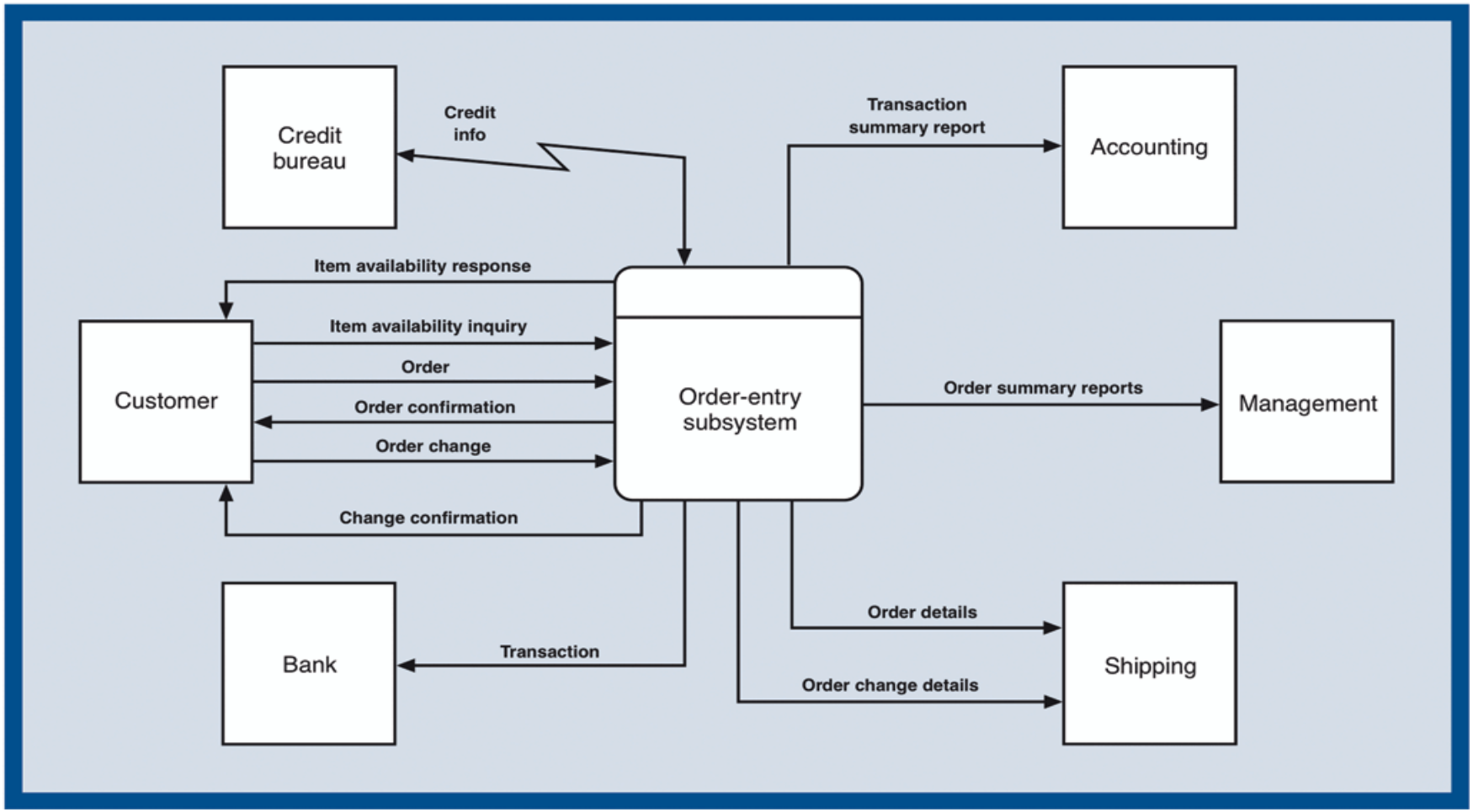


Figure 6-11

RMO Event Table

Customer support system event table					
Event	**Trigger**	**Source**	**Use case**	**Response**	**Destination**
1. Customer wants to check item availability	Item inquiry	Customer	Look up item availability	Item availability details	Customer
2. Customer places an order	New order	Customer	Create new order	Real-time link Order confirmation Order details Transaction	Credit bureau Customer Shipping Bank
3. Customer changes or cancels order	Order change request	Customer	Update order	Change confirmation Order change details Transaction	Customer Shipping Bank
4. Time to produce order summary reports	"End of week, month, quarter and year"		Produce order summary reports	Order summary reports	Management
5. Time to produce transaction summary reports	"End of day"		Produce transaction summary reports	Transaction summary reports	Accounting
6. Customer or management wants to check order status	Order status inquiry	Customer or management	Look up order status	Order status details	Customer or management

Figure 5-12

Five Separate DFD Fragments for RMO Order-Entry Subsystem

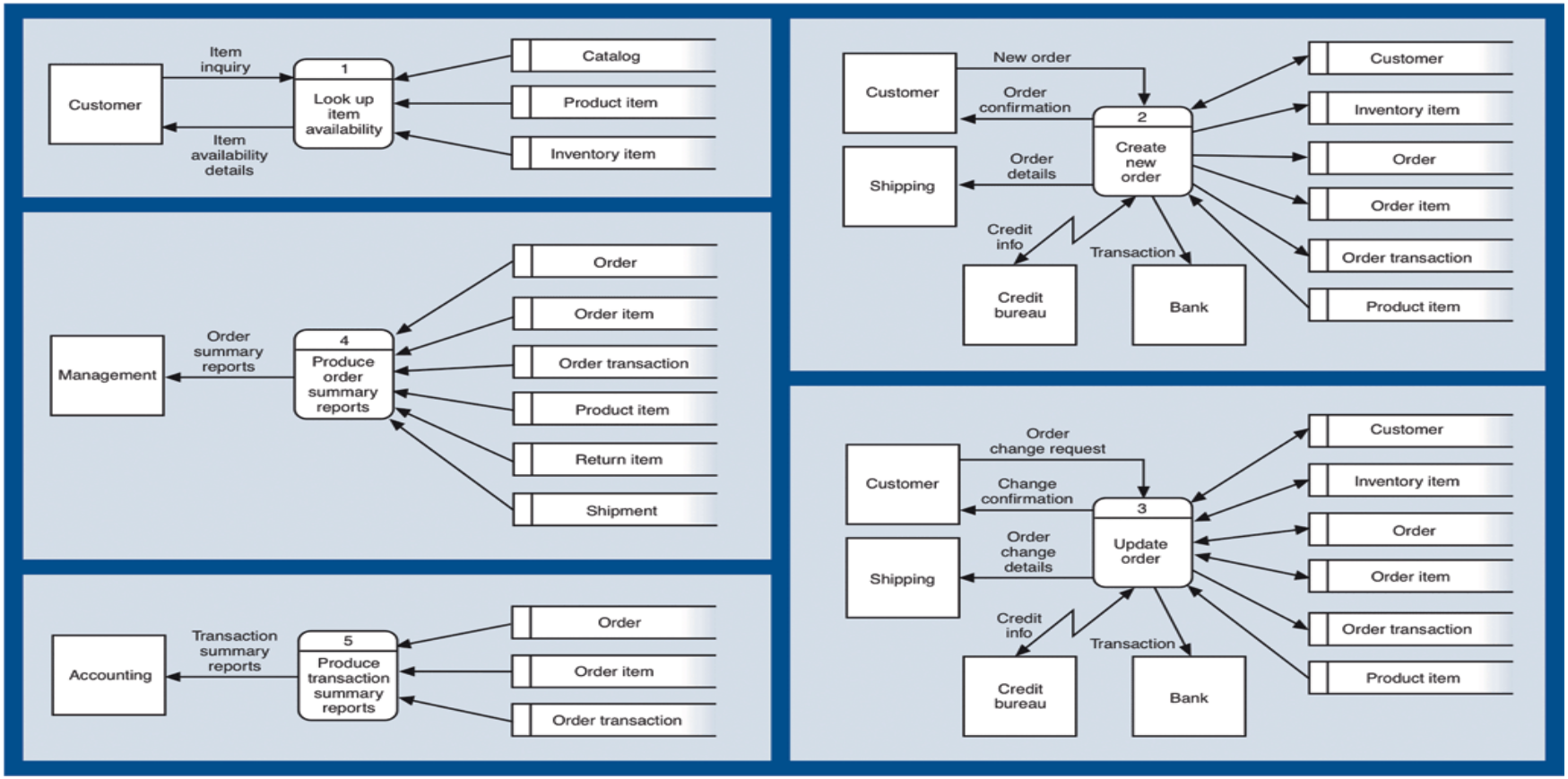


Figure 6-12

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

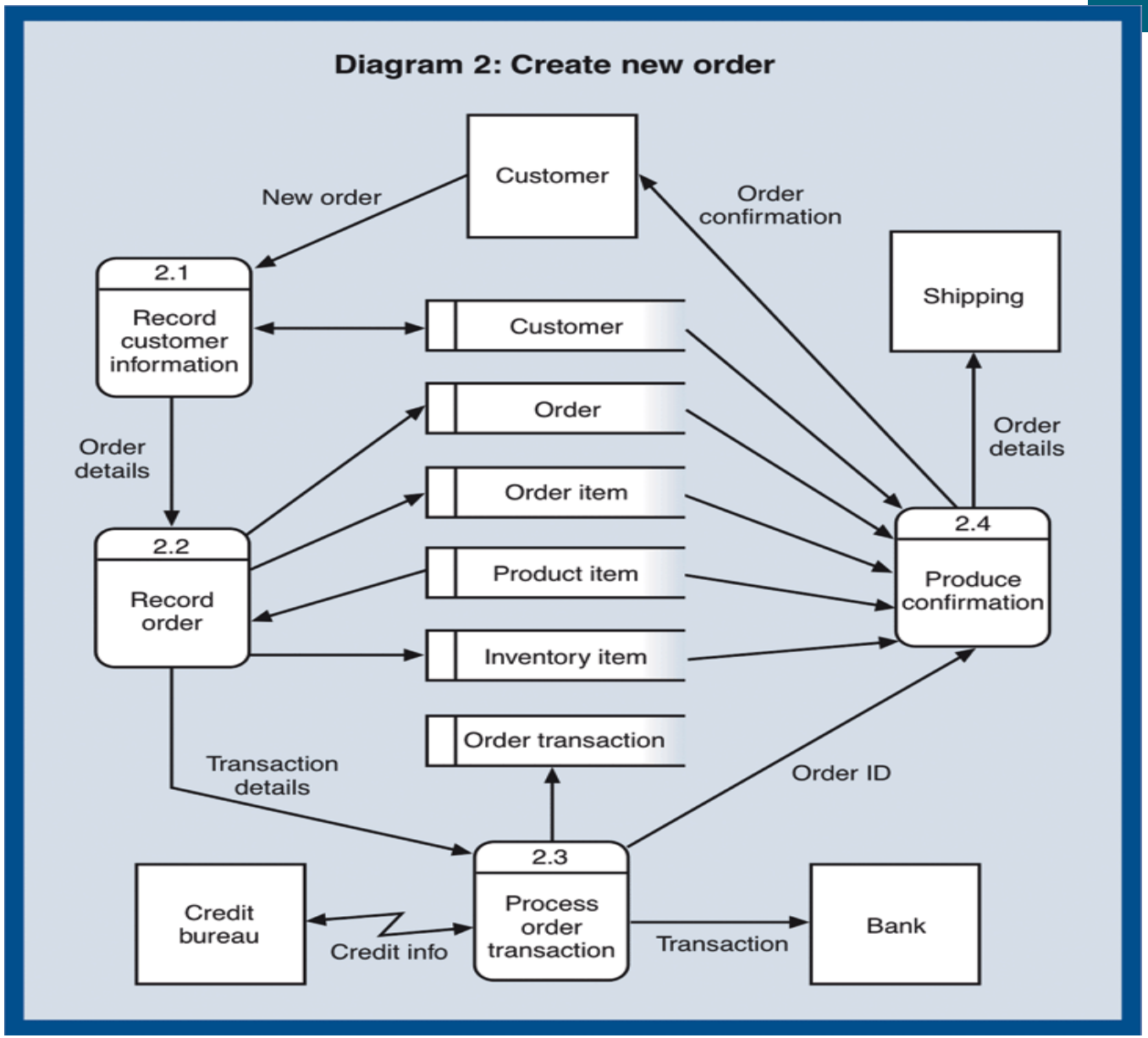


Figure 6-14

Example:
Course Registration System

Event Table

Event	Trigger	Source	Use Case	Response	Destination
User wants to prepare the offered courses	Schedule data	Academic Dept.	Schedule Course	Offered Course	Data Store (Course)
Student wants to register in a course	Enrollment Request	Student	Enroll Student	Schedule	Student
System generates all class lists	Timer	-	Produce Class List	Class List	Faculty Member

Layers of DFD Abstraction for Course Registration System

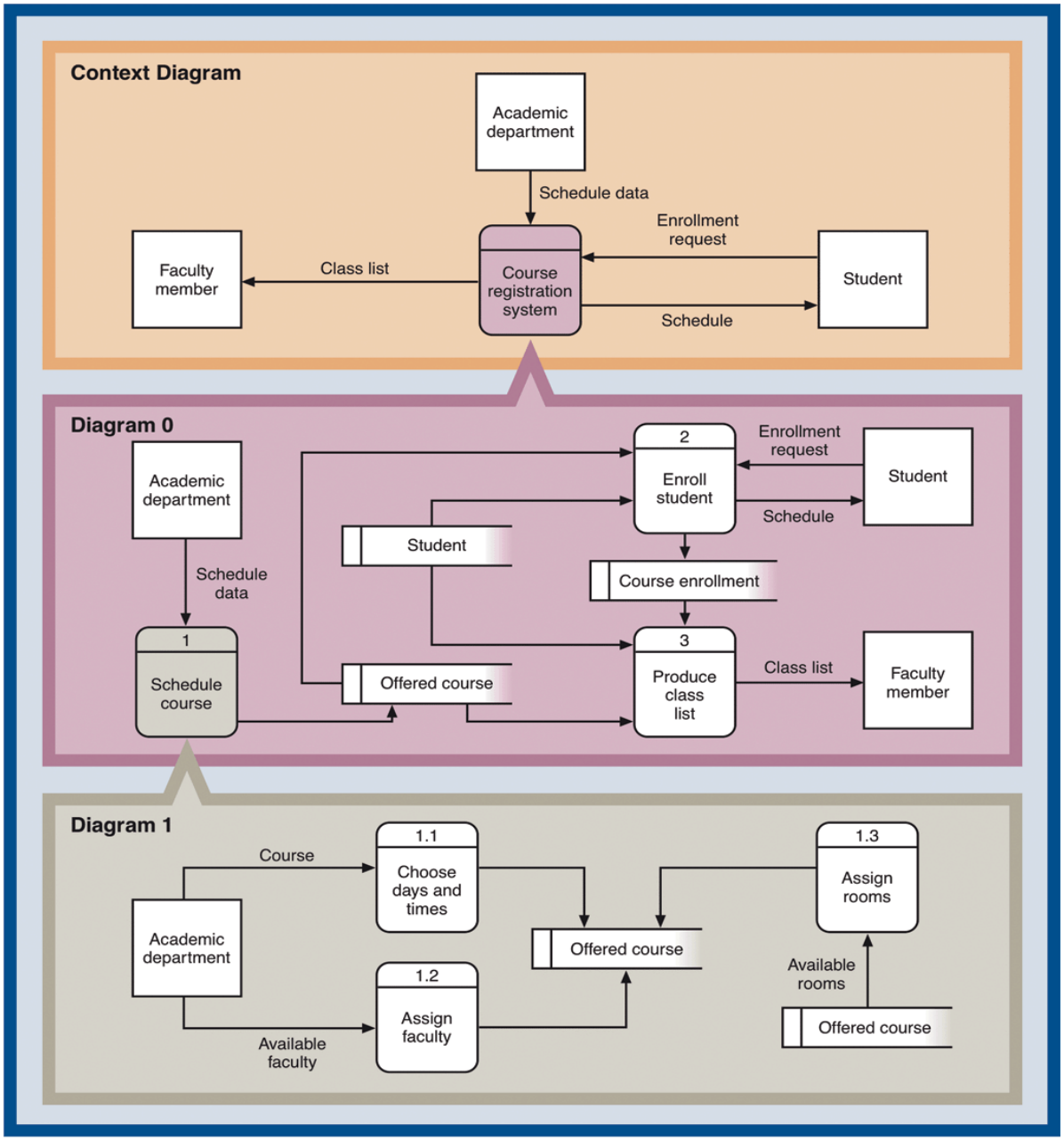


Figure 6-6

Three Separate DFD Fragments for Course Registration System

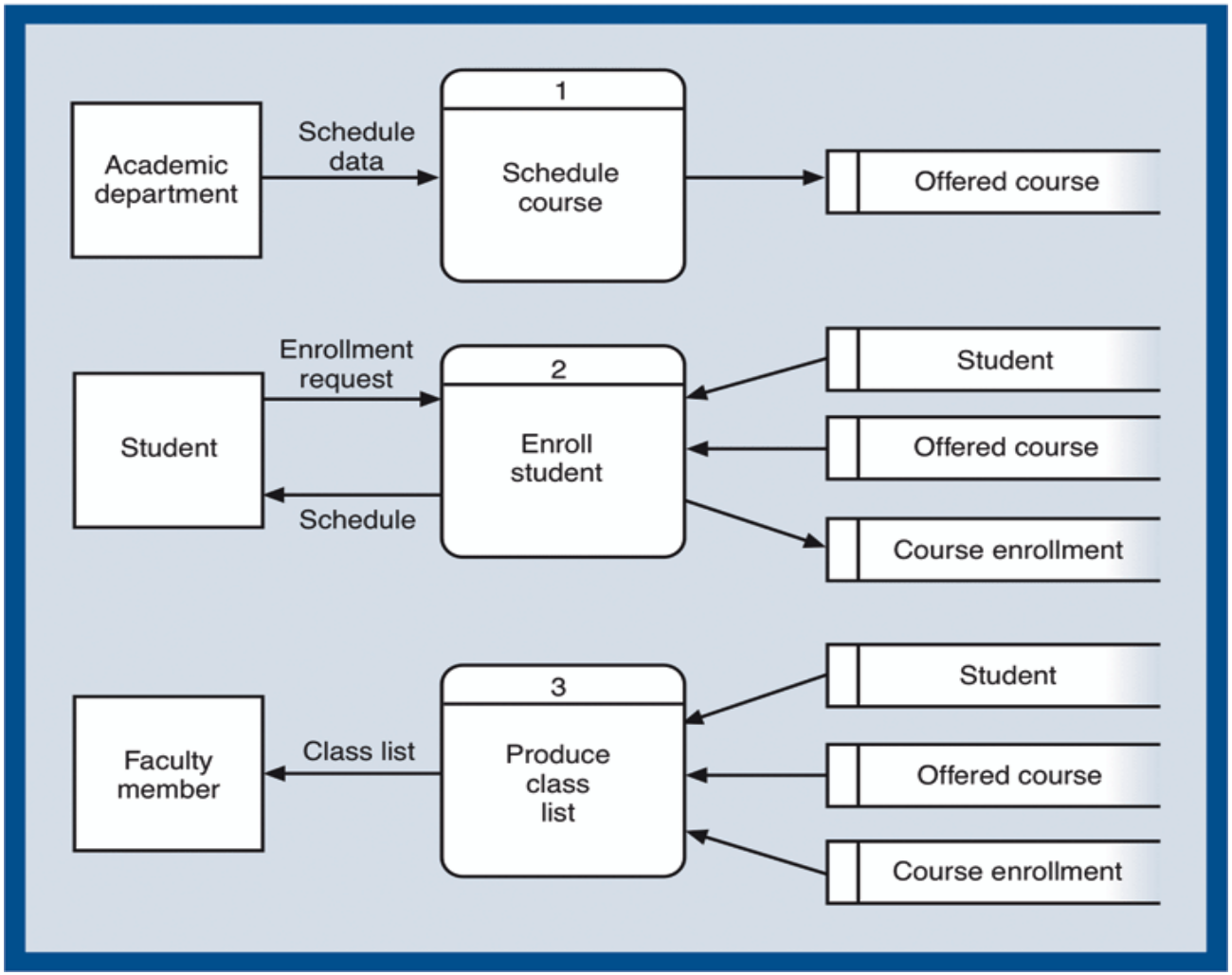


Figure 6-7

Combining DFD Fragments to Create DFD diagram (overview-level)

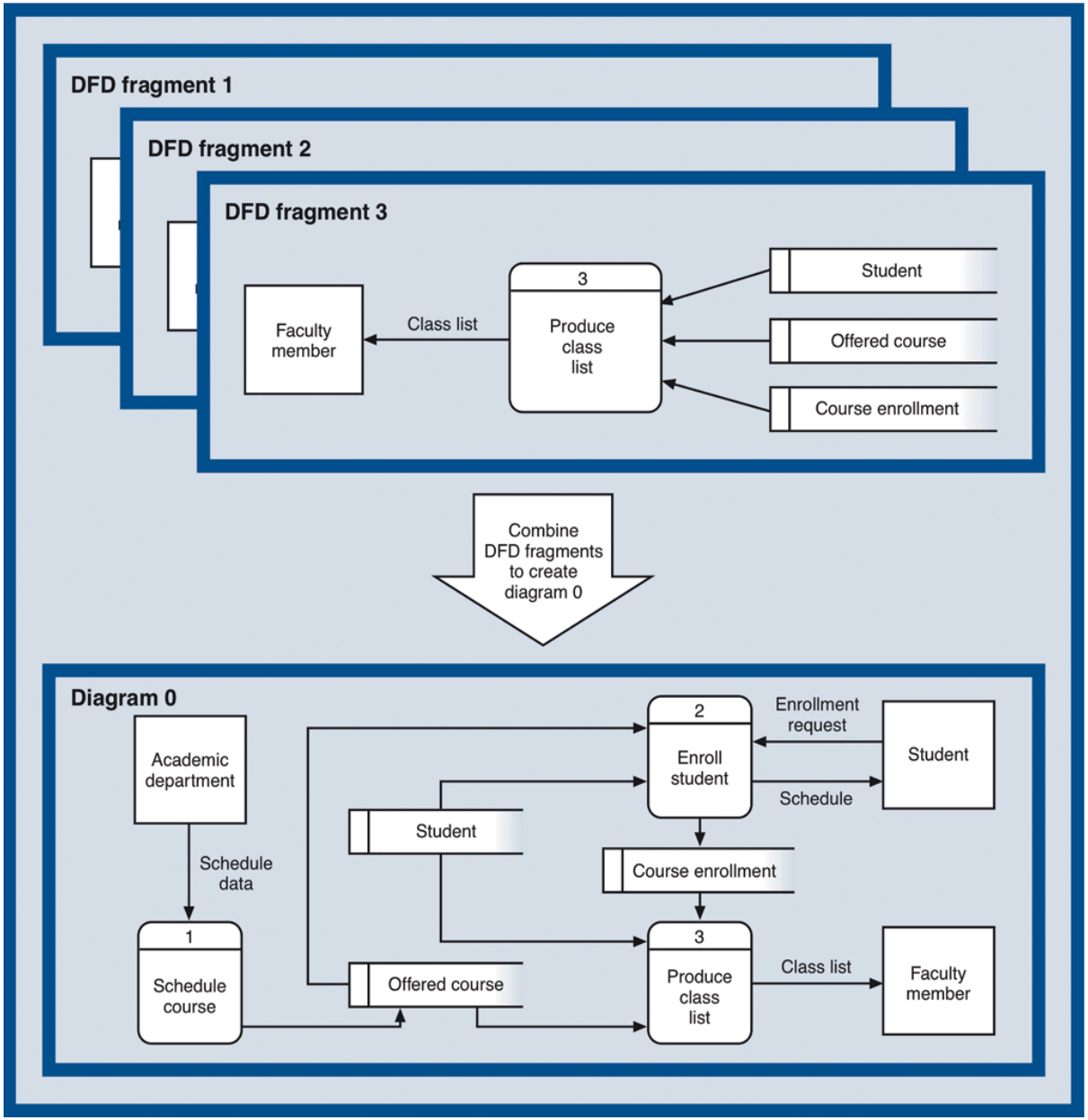


Figure 6-8

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

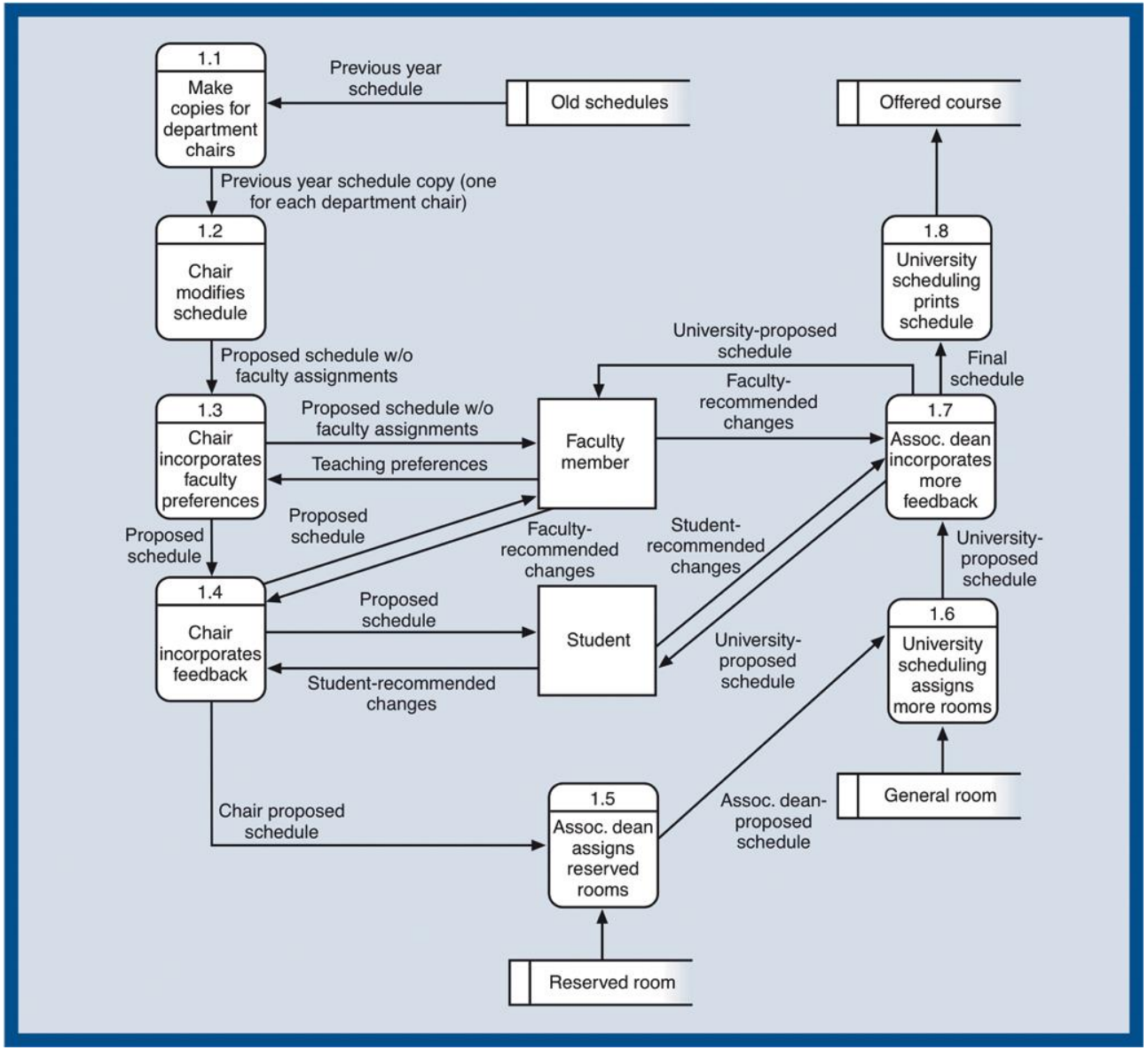


Figure 6-15

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
- Minimum Coupling between main processes
- Maximum Cohesion between sub processes

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

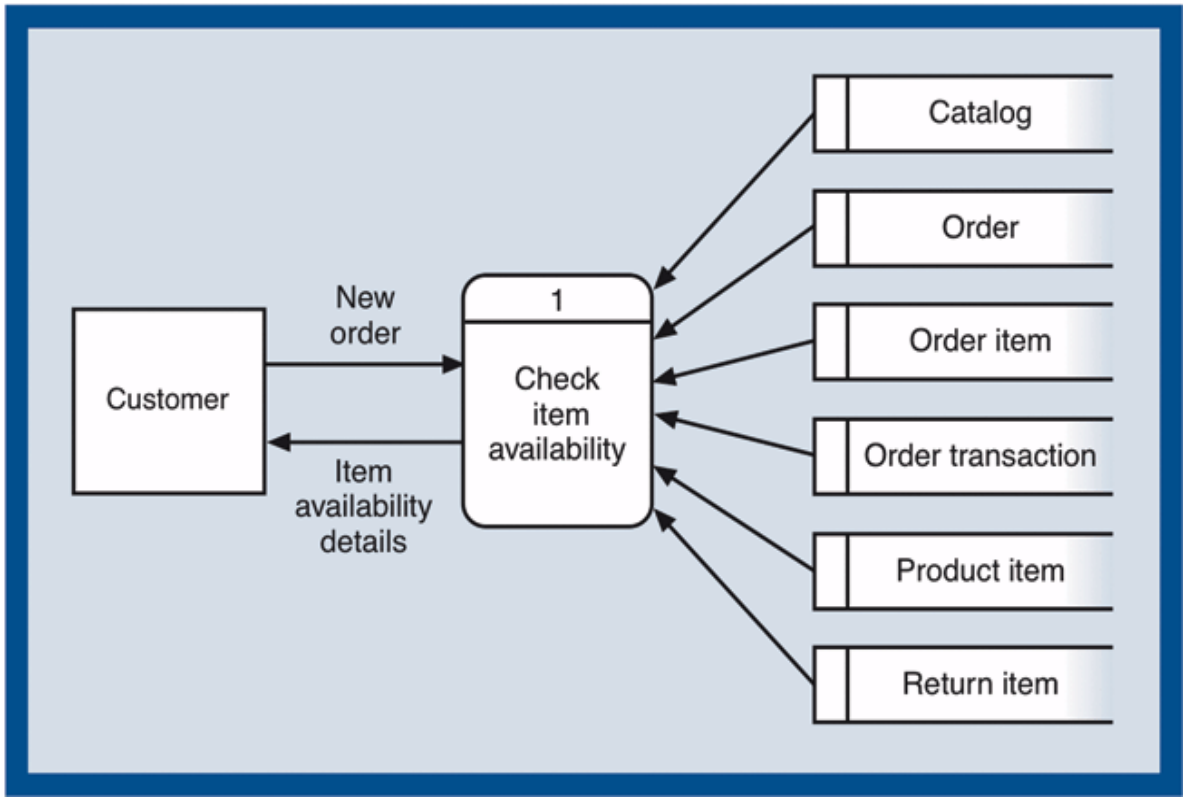


Figure 6-16

Process with Impossible Data Output: a Miracle معجزة

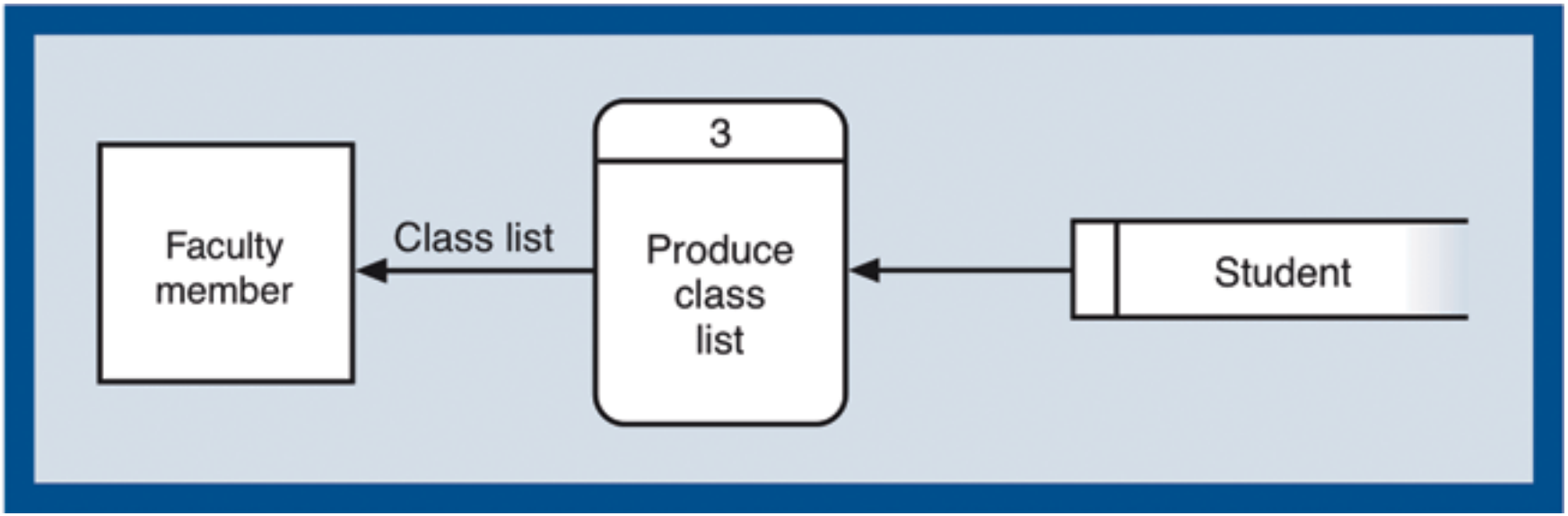


Figure 6-17

Process with Unnecessary Data Input

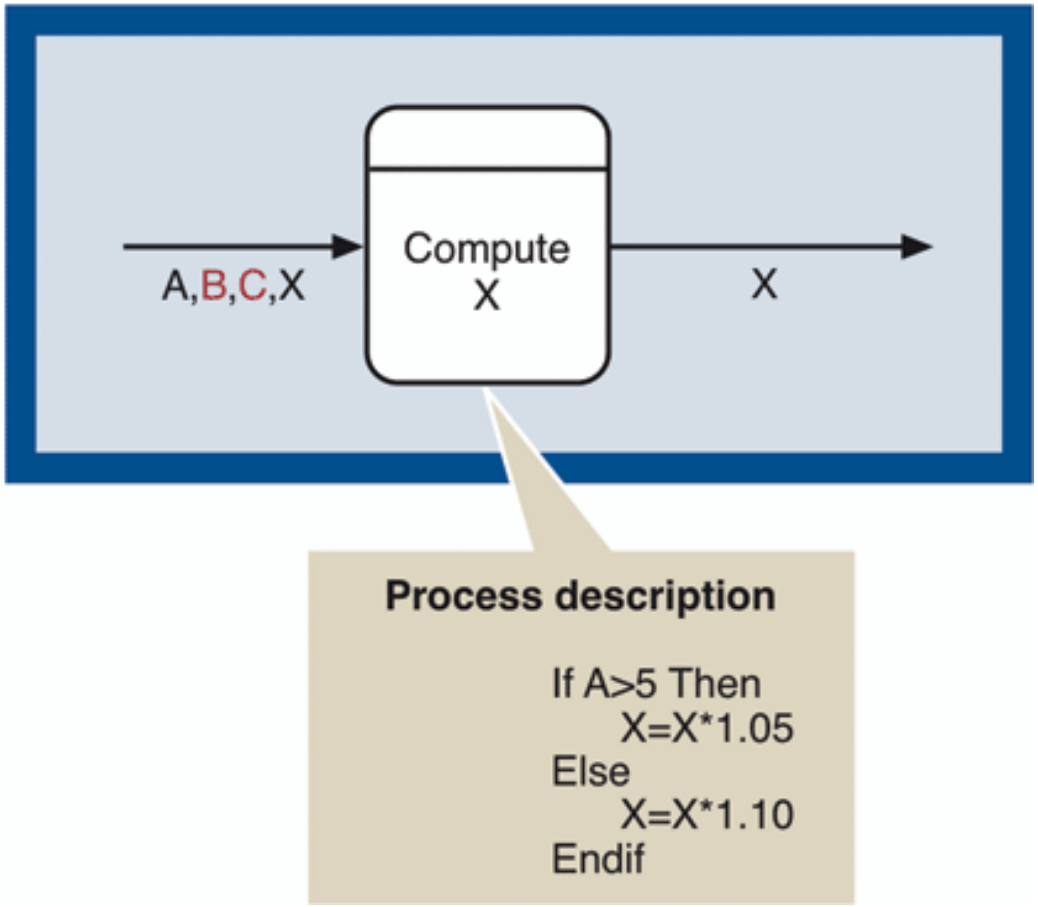


Figure 6-18

Process with Impossible Data Output

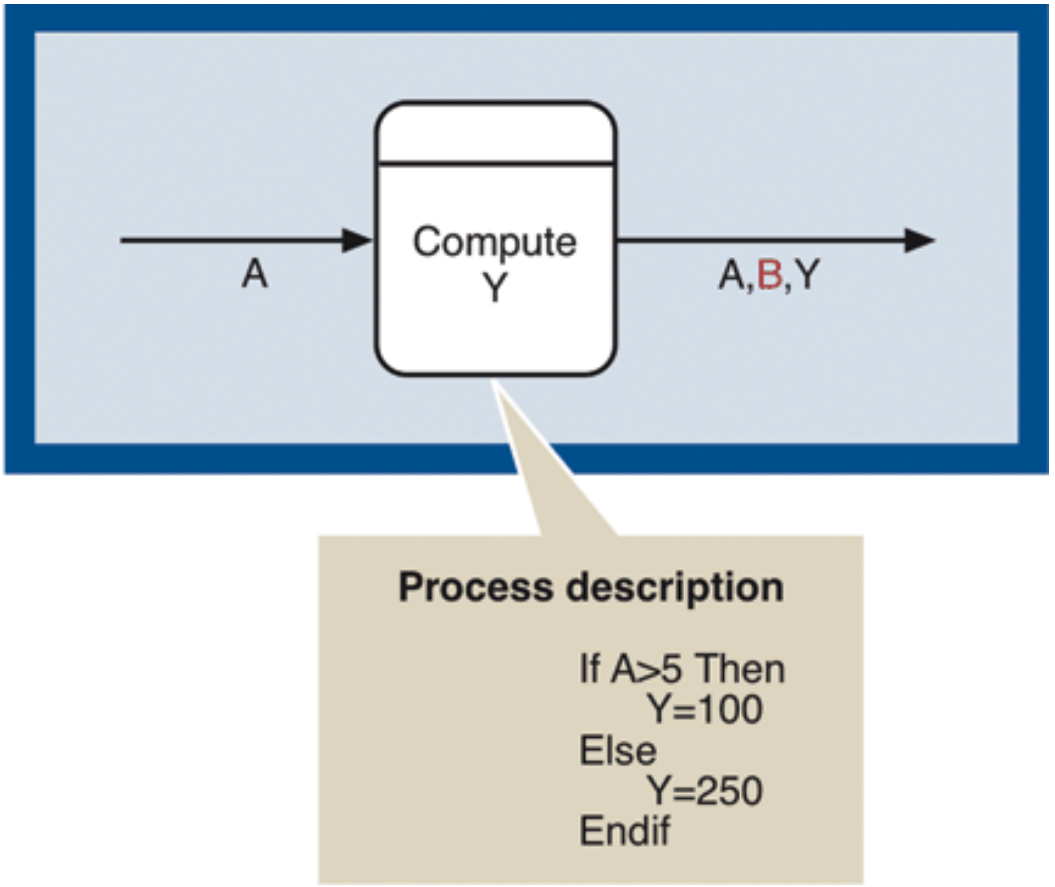


Figure 6-19

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
```

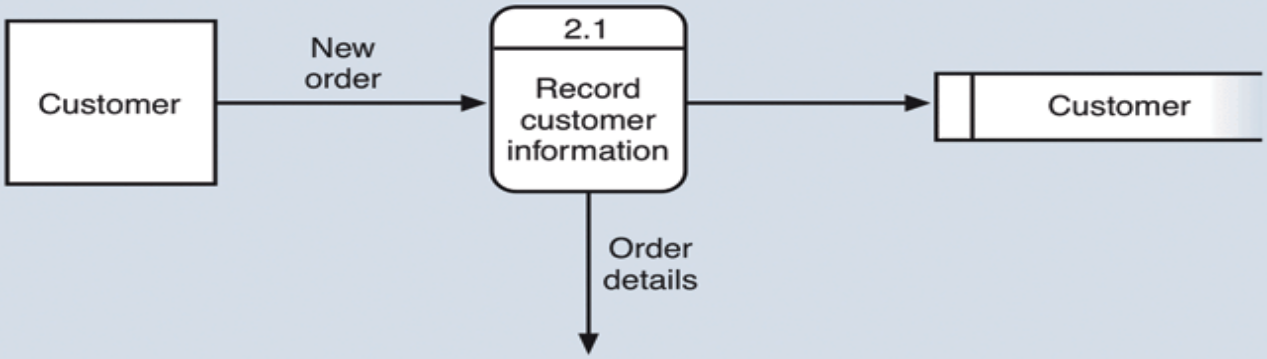


Figure 6-21

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

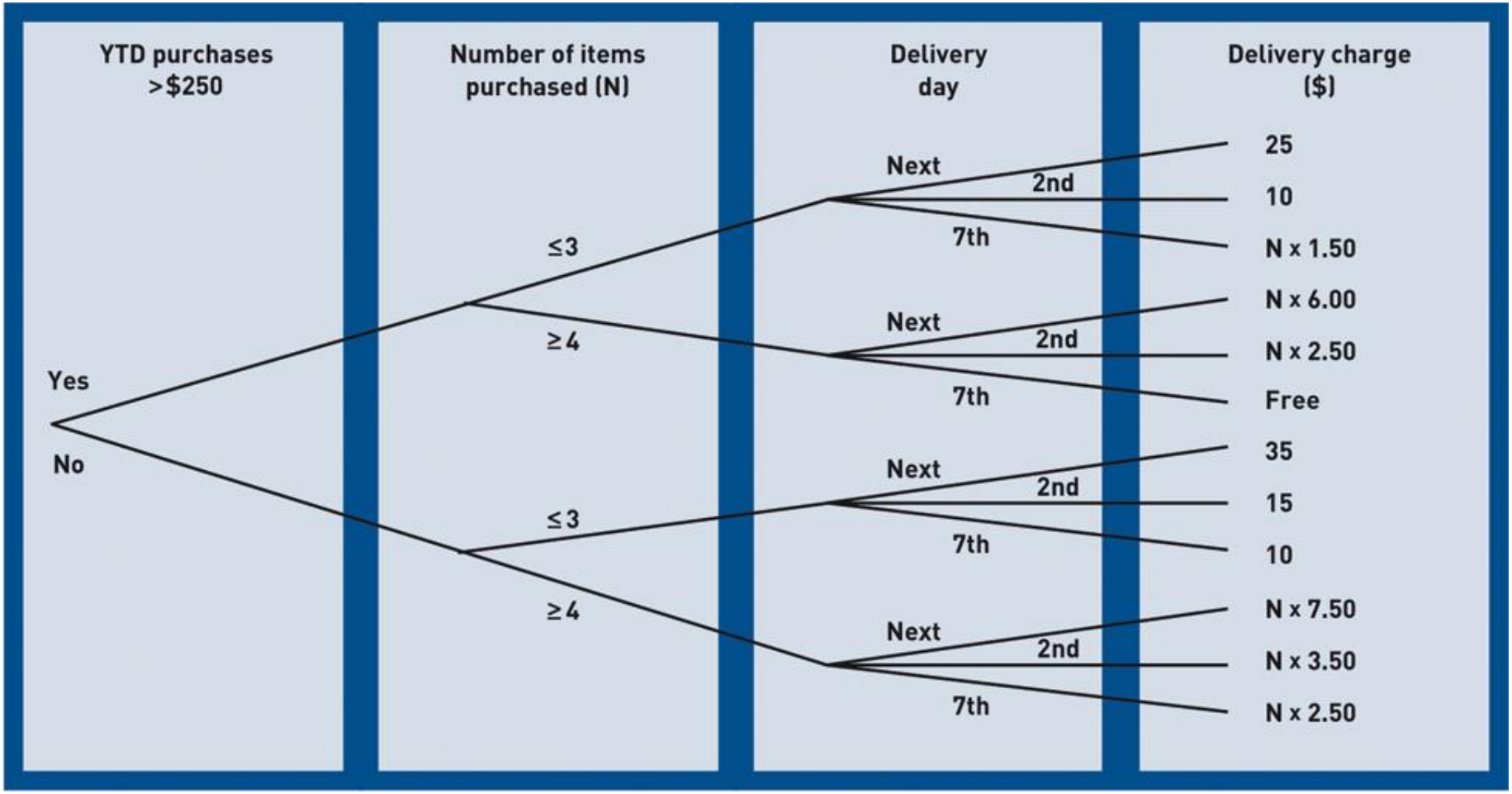


Figure 6-24

Data 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

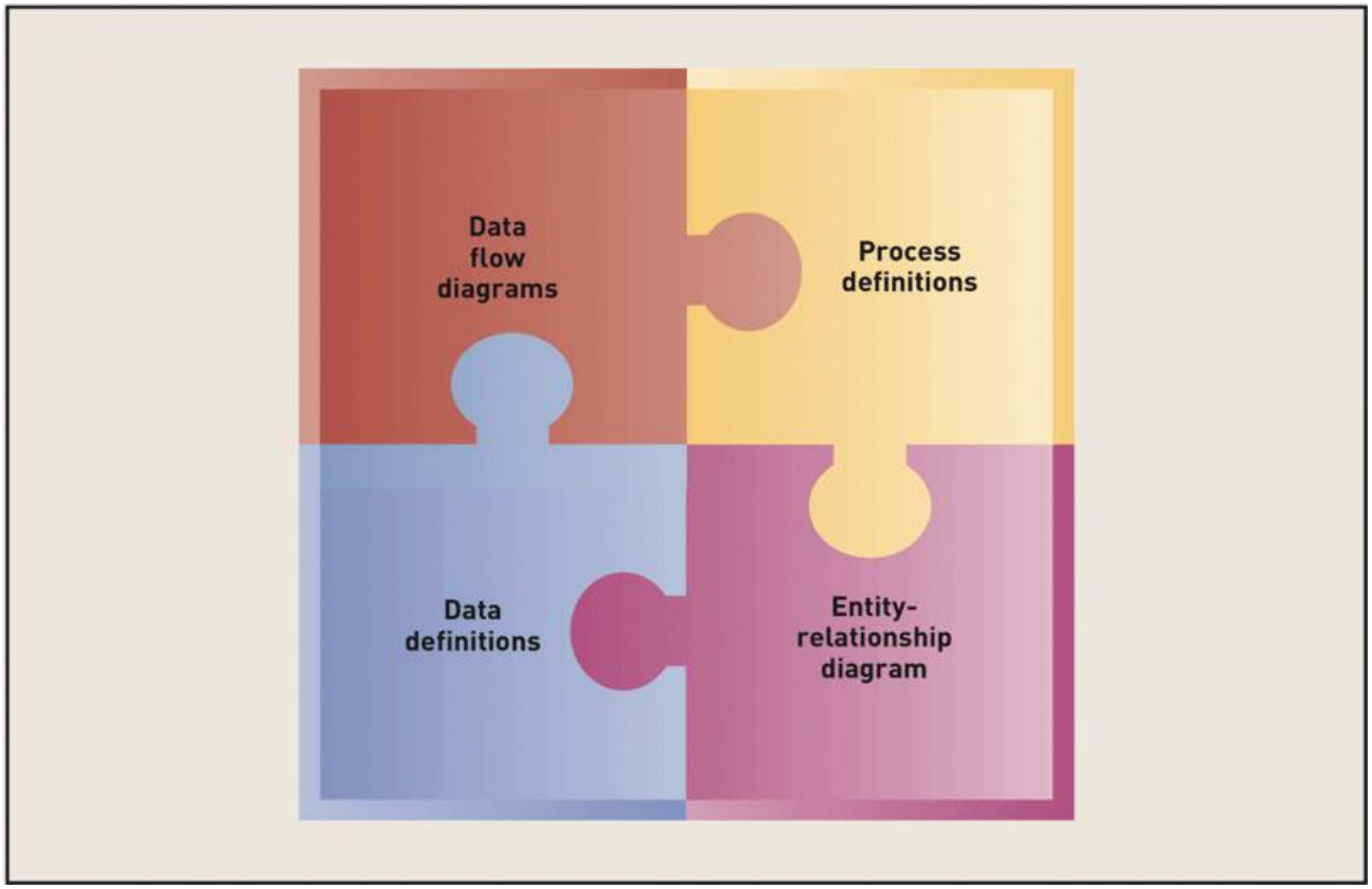


Figure 6-31

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)