Modern Systems Analysis and Design

Chapter 7
Structuring System Process Requirements

Learning Objectives

- ✔ Understand the logical modeling of processes by studying examples of data flow diagrams (DFDs).
- ✔ Draw data flow diagrams following specific rules and guidelines that lead to accurate and well-structured process models.
- ✔ Decompose data flow diagrams into lower-level diagrams.
- ✔ Balance higher-level and lower-level data flow diagrams.

Learning Objectives (Cont.)

- ✓ Use data flow diagrams as a tool to support the analysis of information systems.
- ✓ Discuss process modeling for electronic commerce applications.
- ✓ Use decision tables to represent the logic of choice in conditional statements.

Process Modeling

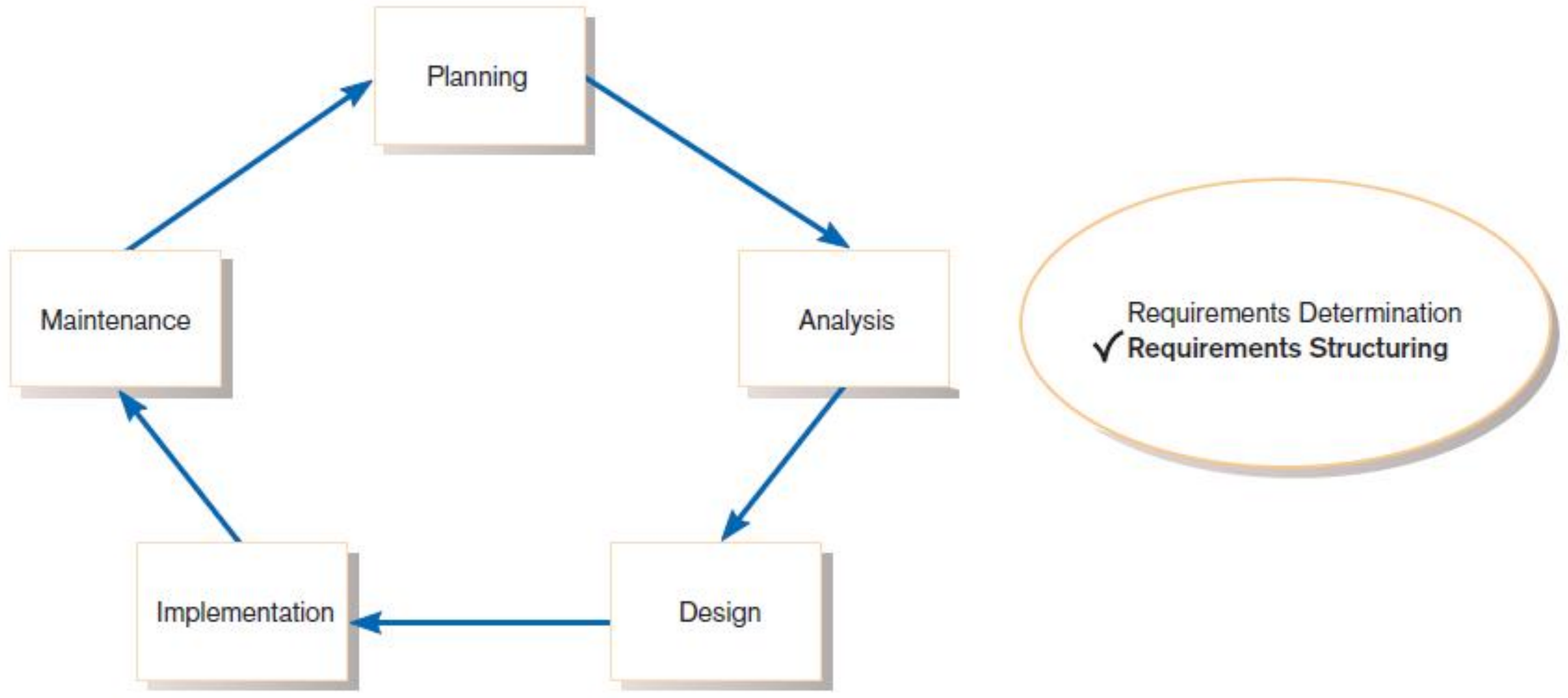


FIGURE 7-1
Systems development life cycle with the analysis phase highlighted

Process Modeling (Cont.)

- Graphically represent the processes that capture, manipulate, store, and distribute data between a system and its environment and among system components.

Process Modeling (Cont.)

- Utilize information gathered during requirements determination.
- Processes and data structures are modeled.

Deliverables and Outcomes

- Context data flow diagram (DFD)
  - Scope of system
- DFDs of current physical system
  - Adequate detail only
- DFDs of current logical system
  - Enables analysts to understand current system

Deliverables and Outcomes (Cont.)

- DFDs of new logical system
  - Technology independent
  - Show data flows, structure, and functional requirements of new system
- Thorough description of each DFD component

Data Flow Diagramming Mechanics

- Represent both physical and logical information systems
- Only four symbols are used

Data Flow Diagramming Mechanics (Cont.)

- Useful for depicting purely logical information flows
- DFDs that detail physical systems differ from system flowcharts which depict details of physical computing equipment

Definitions and Symbols

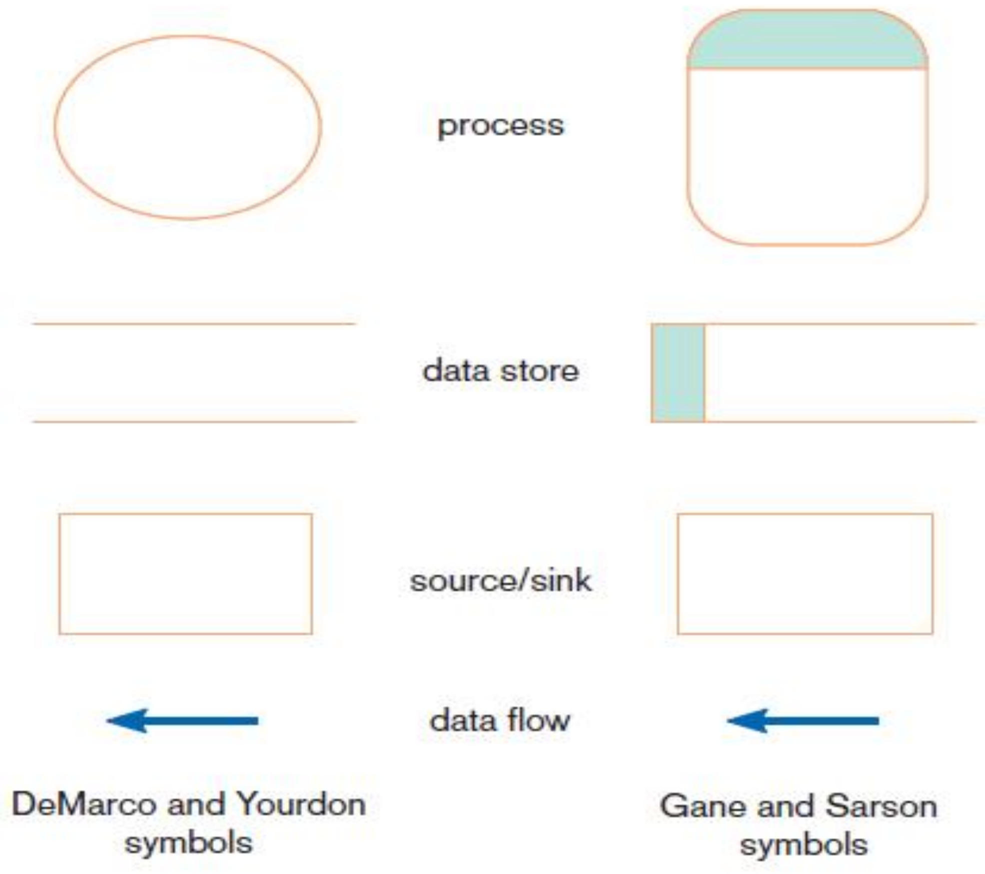


FIGURE 7-2
Comparison of DeMarco and Yourdon
and Gane and Sarson
DFD symbol sets

Definitions and Symbols (Cont.)

- **Process**: work or actions performed on data (inside the system)
- **Data store**: data at rest (inside the system)

Definitions and Symbols (Cont.)

- **Source/sink**: external entity that is origin or destination of data (outside the system)
- **Data flow**: arrows depicting movement of data

Developing DFDs

- **Context diagram** is an overview of an organizational system that shows:
  - the system boundaries.
  - external entities that interact with the system.
  - major information flows between the entities and the system.
- Note: only one process symbol, and no data stores shown

Context Diagram

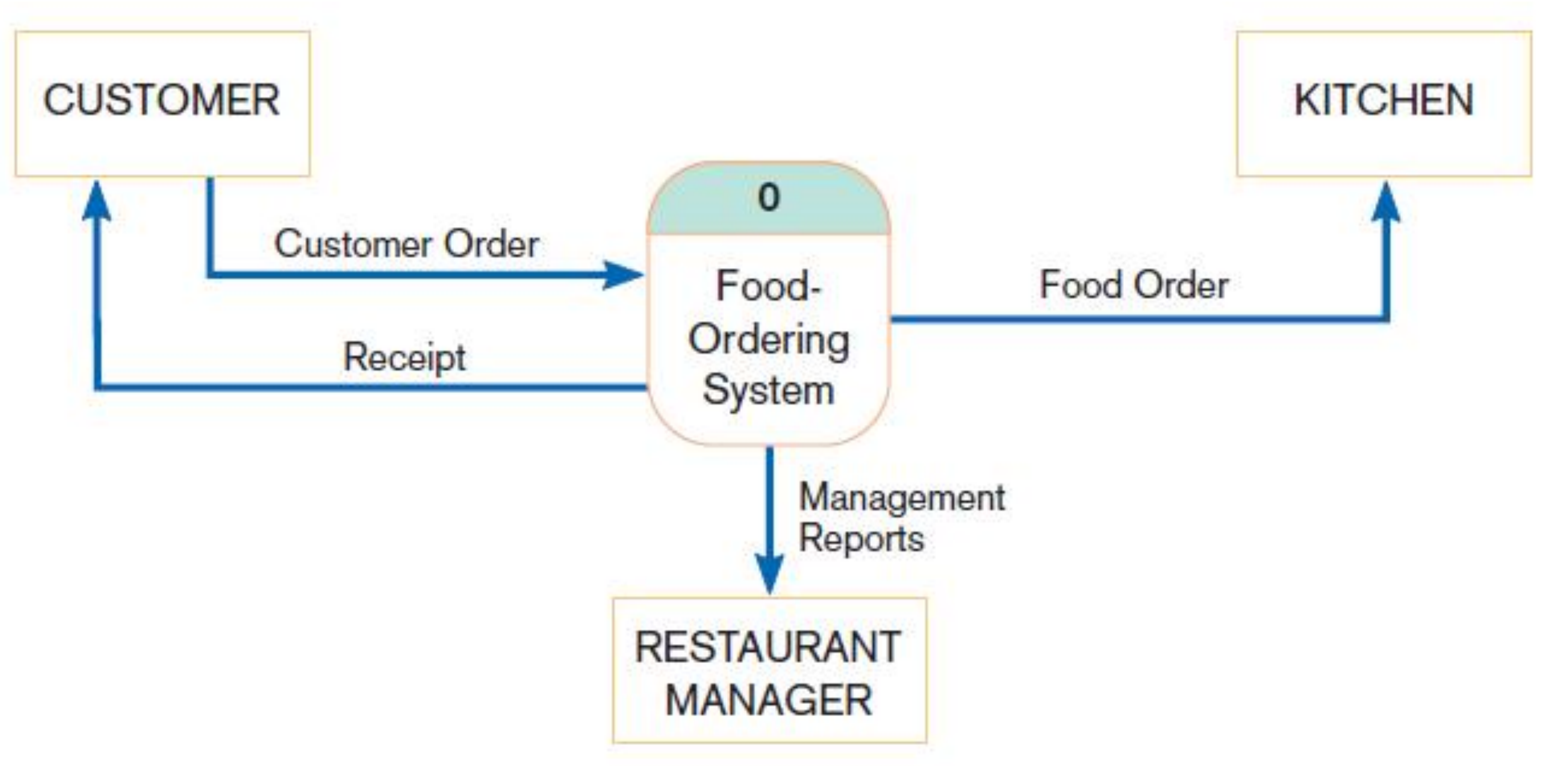


FIGURE 7-4
Context diagram of Hoosier Burger's food-ordering system

Developing DFDs (Cont.)

- **Level-0 diagram** is a data flow diagram that represents a system’s major processes, data flows, and data stores at a high level of detail.
    - Processes are labeled 1.0, 2.0, etc. These will be decomposed into more primitive (lower-level) DFDs.

Level-0 Diagram

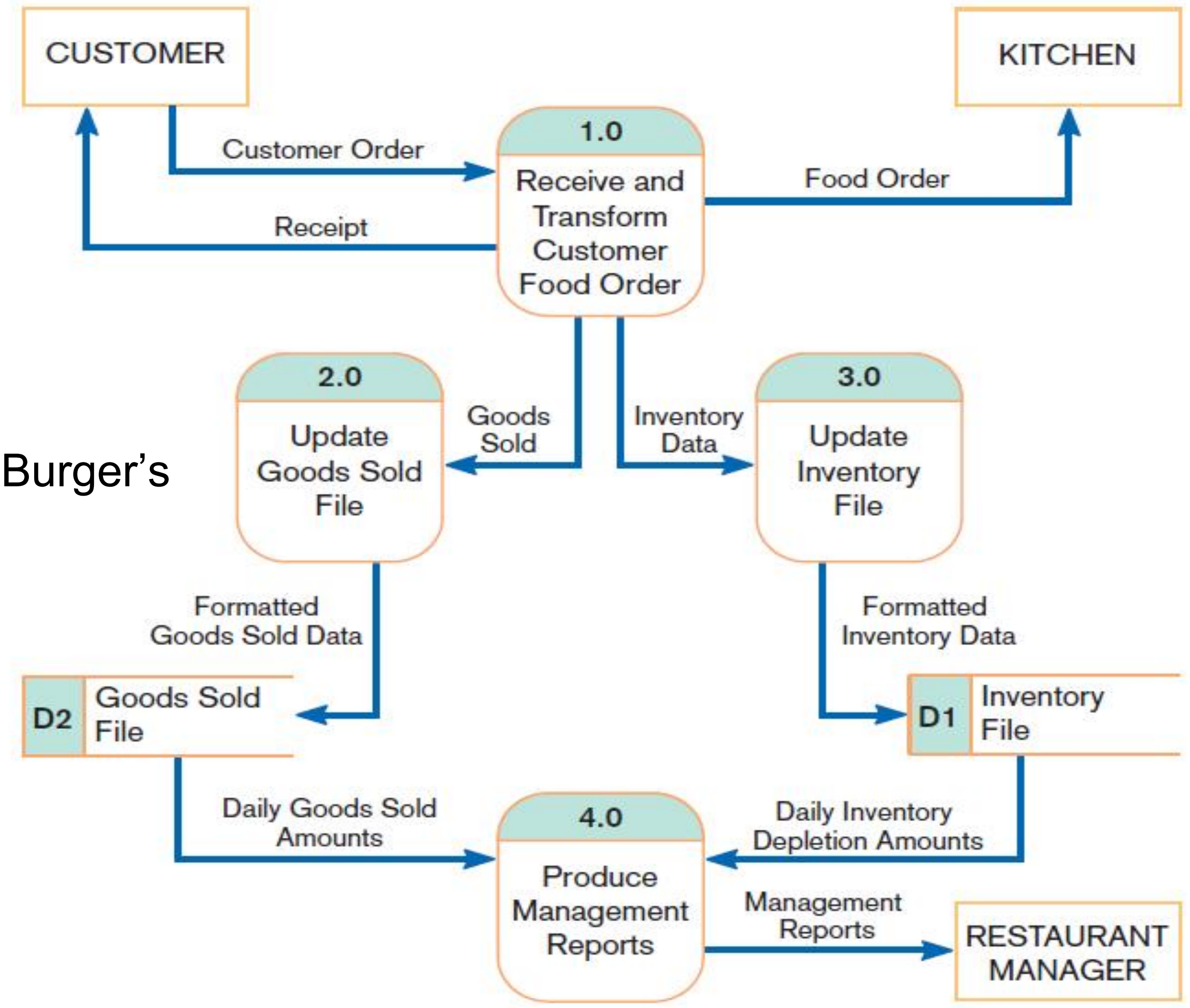


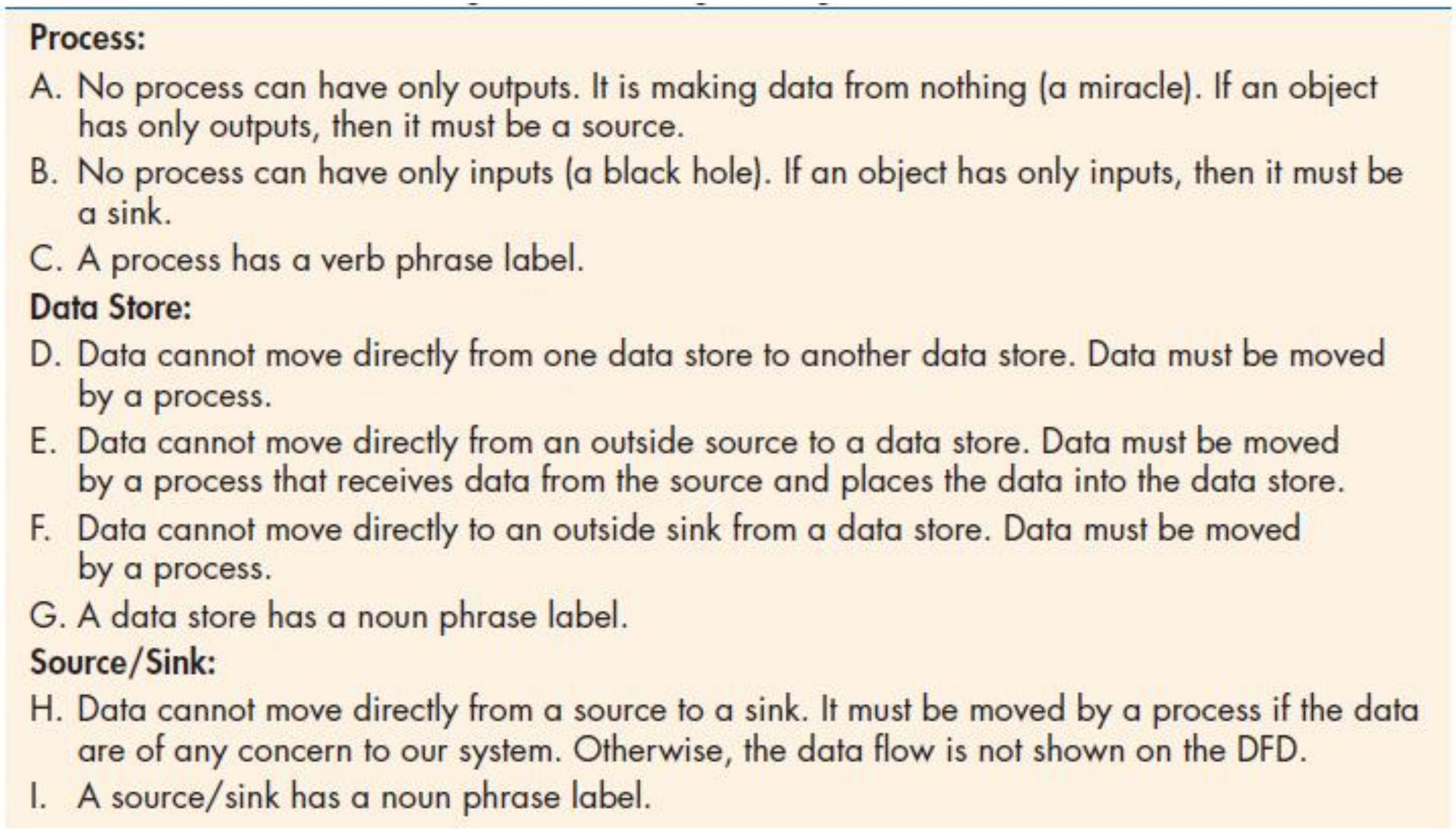
FIGURE 7-5
Level-0 DFD of Hoosier Burger's food-ordering system

Data Flow Diagramming Rules

- There are two DFD guidelines that apply:
  - *The inputs to a process are different from the outputs of that process.*
    - Processes purpose is to transform inputs into outputs.
  - *Objects on a DFD have unique names.*
    - Every process has a unique name.

Data Flow Diagramming Rules (Cont.)

TABLE 7-2 Rules Governing Data Flow Diagramming



Process:

A. No process can have only outputs. It is making data from nothing (a miracle). If an object has only outputs, then it must be a source.

B. No process can have only inputs (a black hole). If an object has only inputs, then it must be a sink.

C. A process has a verb phrase label.

Data Store:

D. Data cannot move directly from one data store to another data store. Data must be moved by a process.

E. Data cannot move directly from an outside source to a data store. Data must be moved by a process that receives data from the source and places the data into the data store.

F. Data cannot move directly to an outside sink from a data store. Data must be moved by a process.

G. A data store has a noun phrase label.

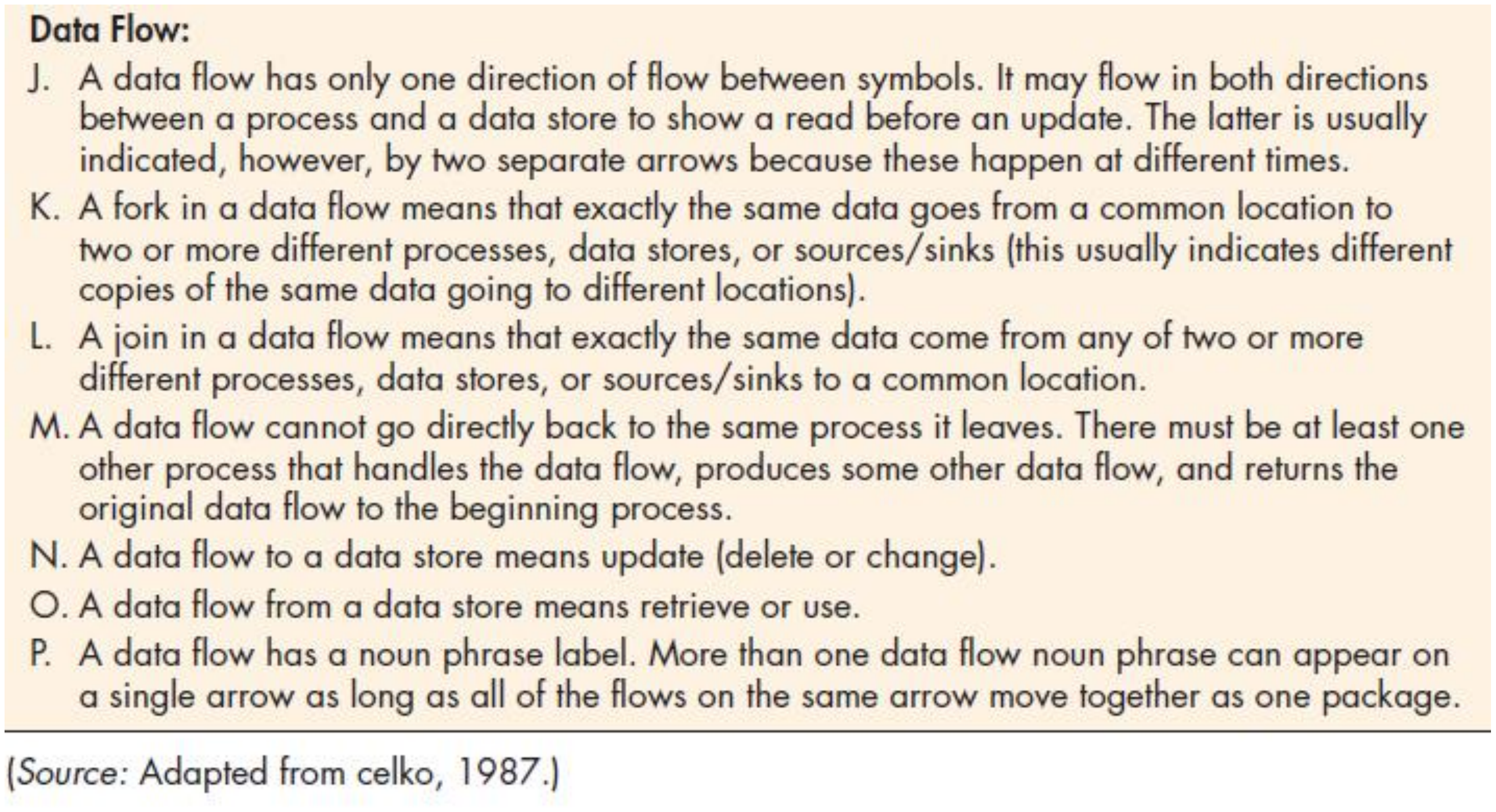
Source/Sink:

H. Data cannot move directly from a source to a sink. It must be moved by a process if the data are of any concern to our system. Otherwise, the data flow is not shown on the DFD.

I. A source/sink has a noun phrase label.

Data Flow Diagramming Rules (Cont.)

TABLE 7-2 Rules Governing Data Flow Diagramming (cont.)



Data Flow:

J. A data flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated, however, by two separate arrows because these happen at different times.

K. A fork in a data flow means that exactly the same data goes from a common location to two or more different processes, data stores, or sources/sinks (this usually indicates different copies of the same data going to different locations).

L. A join in a data flow means that exactly the same data come from any of two or more different processes, data stores, or sources/sinks to a common location.

M. A data flow cannot go directly back to the same process it leaves. There must be at least one other process that handles the data flow, produces some other data flow, and returns the original data flow to the beginning process.

N. A data flow to a data store means update (delete or change).

O. A data flow from a data store means retrieve or use.

P. A data flow has a noun phrase label. More than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

(*Source:* Adapted from celko, 1987.)

Decomposition of DFDs

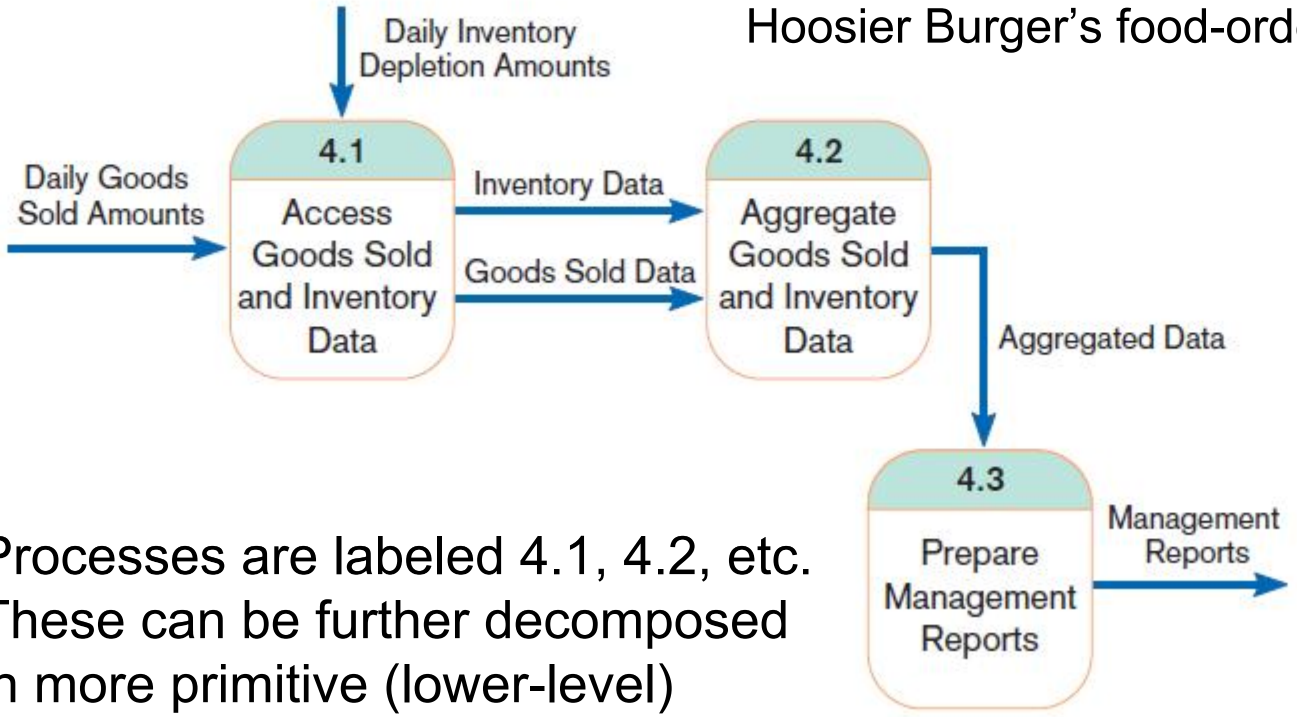
- **Functional decomposition** is an iterative process of breaking a system description down into finer and finer detail.
  - Creates a set of charts in which one process on a given chart is explained in greater detail on another chart.
  - Continues until no subprocess can logically be broken down any further.

Decomposition of DFDs (Cont.)

- *Primitive DFD* is the lowest level of a DFD.
- **Level-1 diagram** results from decomposition of Level-0 diagram.
- **Level-n diagram** is a DFD diagram that is the result of ***n*** nested decompositions from a process on a level-0 diagram.

Level-1 DFD

FIGURE 7-8
Level-1 diagram showing the decomposition of Process 4.0 from the level-0 diagram for Hoosier Burger's food-ordering system


Daily Inventory Depletion Amounts
Daily Goods Sold Amounts
4.1
Access Goods Sold and Inventory Data
Inventory Data
Goods Sold Data
4.2
Aggregate Goods Sold and Inventory Data
Aggregated Data
4.3
Prepare Management Reports
Management Reports

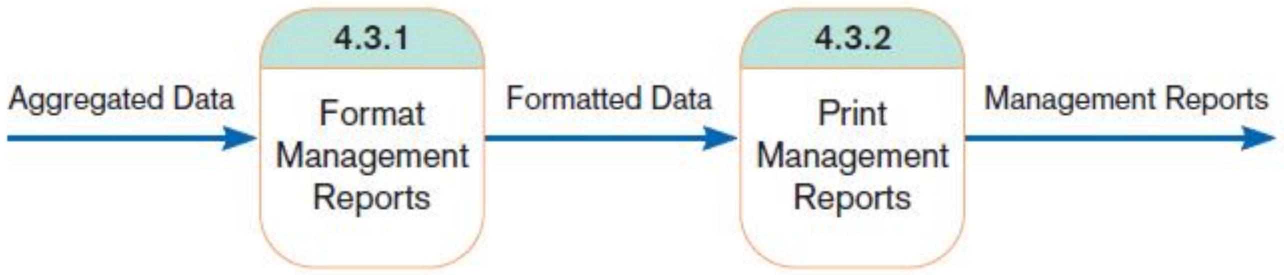
Level-1 DFD shows the sub-processes of one of the processes in the Level-0 DFD.

This is a Level-1 DFD for Process 4.0.

Processes are labeled 4.1, 4.2, etc. These can be further decomposed in more primitive (lower-level) DFDs if necessary.

Level-*n* DFD

FIGURE 7-9
Level-2 diagram showing the decomposition of Process 4.3 from the level-1 diagram for Process 4.0 for Hoosier Burger's food-ordering system



Level-*n* DFD shows the sub-processes of one of the processes in the Level *n-1* DFD.

This is a Level-2 DFD for Process 4.3.

Processes are labeled 4.3.1, 4.3.2, etc. If this is the lowest level of the hierarchy, it is called a *primitive DFD.*

Balancing DFDs

- **Conservation Principle**: conserve inputs and outputs to a process at the next level of decomposition
- **Balancing:** conservation of inputs and outputs to a data flow diagram process when that process is decomposed to a lower level

Balancing DFDs (Cont.)

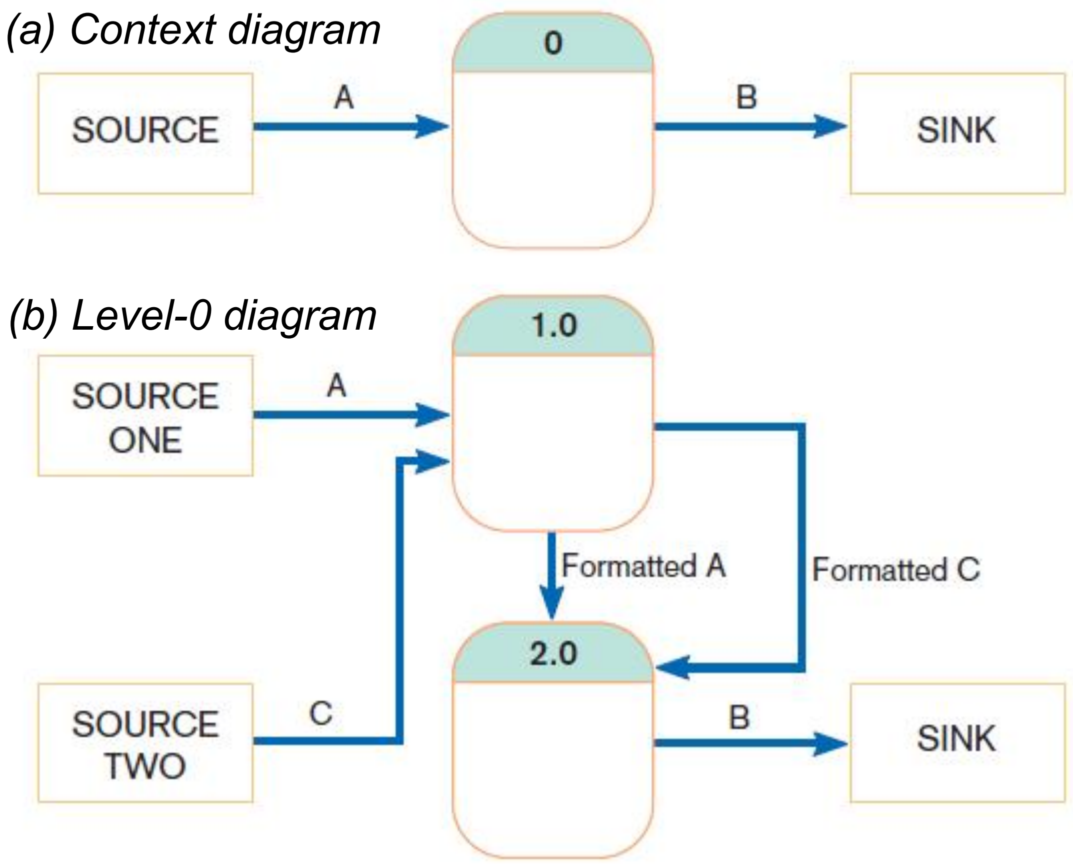
- Balanced means:
  - Number of inputs to lower level DFD equals number of inputs to associated process of higher-level DFD
  - Number of outputs to lower level DFD equals number of outputs to associated process of higher-level DFD

Balancing DFDs (Cont.)

FIGURE 7-10 An unbalanced set of data flow diagrams

1 input
1 output

2 inputs
1 output



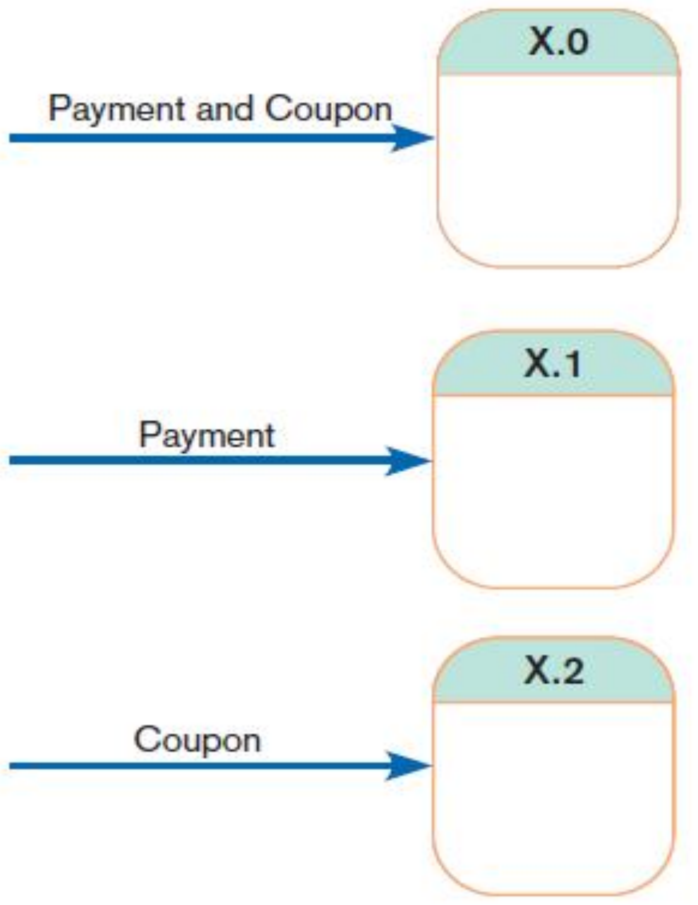
This is unbalanced because the process of the context diagram has only one input but the Level-0 diagram has two inputs.

Balancing DFDs (Cont.)

- **Data flow splitting** is when a composite data flow at a higher level is split and different parts go to different processes in the lower level DFD.
- The DFD remains balanced because the same data is involved, but split into two parts.

Balancing DFDs (Cont.)

FIGURE 7-11
Example of data flow splitting



(a) Composite data flow

(b) Disaggregated data flows

Balancing DFDs: More DFD Rules

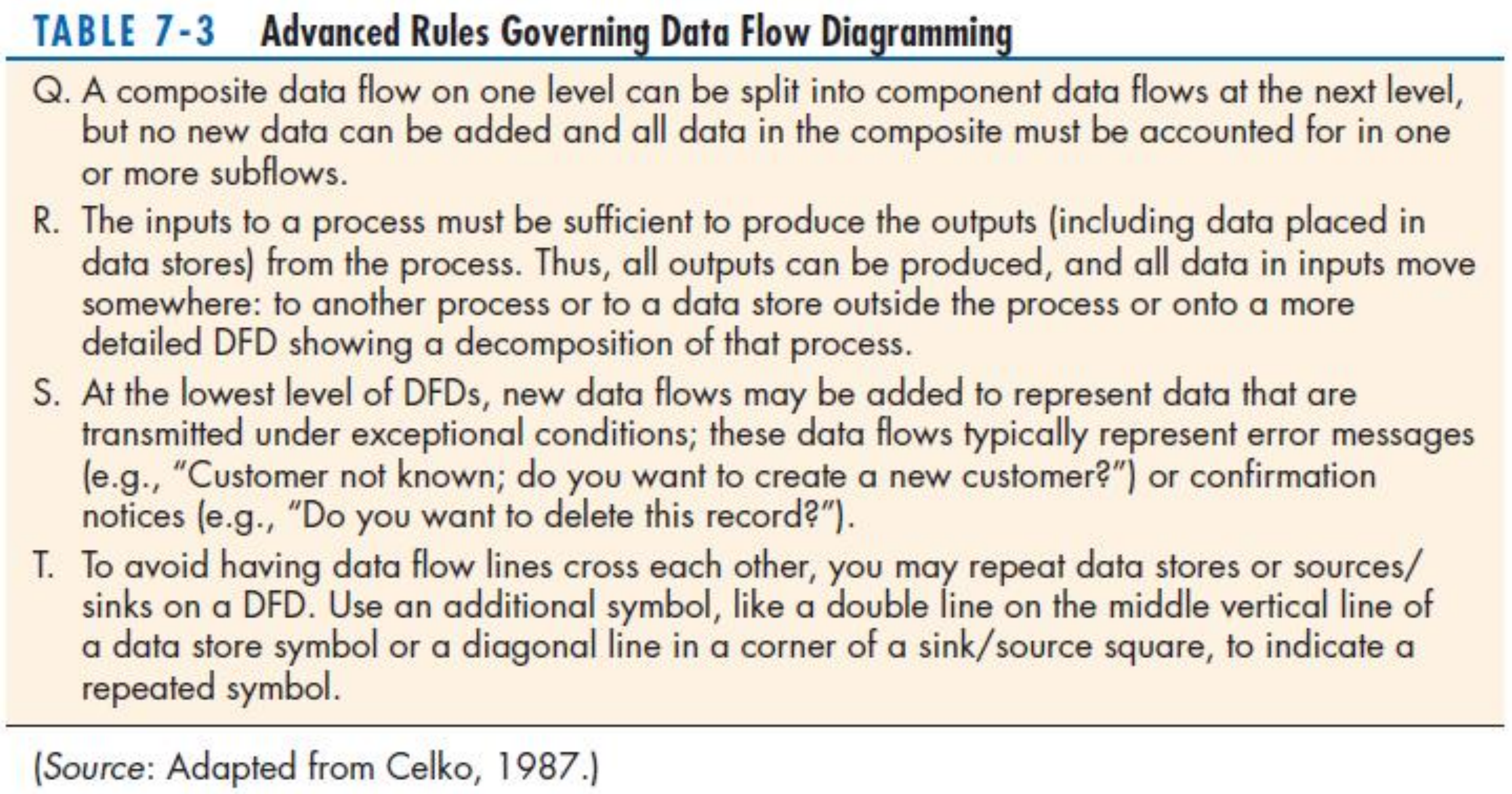


TABLE 7-3 Advanced Rules Governing Data Flow Diagramming

Q. A composite data flow on one level can be split into component data flows at the next level, but no new data can be added and all data in the composite must be accounted for in one or more subflows.

R. The inputs to a process must be sufficient to produce the outputs (including data placed in data stores) from the process. Thus, all outputs can be produced, and all data in inputs move somewhere: to another process or to a data store outside the process or onto a more detailed DFD showing a decomposition of that process.

S. At the lowest level of DFDs, new data flows may be added to represent data that are transmitted under exceptional conditions; these data flows typically represent error messages (e.g., "Customer not known; do you want to create a new customer?") or confirmation notices (e.g., "Do you want to delete this record?").

T. To avoid having data flow lines cross each other, you may repeat data stores or sources/sinks on a DFD. Use an additional symbol, like a double line on the middle vertical line of a data store symbol or a diagonal line in a corner of a sink/source square, to indicate a repeated symbol.

(*Source*: Adapted from Celko, 1987.)

Four Different Types of DFDs

- **Current Physical**
  - Process labels identify technology (people or systems) used to process the data.
  - Data flows and data stores identify actual name of the physical media.
- **Current Logical**
  - Physical aspects of system are removed as much as possible.
  - Current system is reduced to data and processes that transform them.

Four Different Types of DFDs (Cont.)

- **New Logical**
  - Includes additional functions.
  - Obsolete functions are removed.
  - Inefficient data flows are reorganized.
- **New Physical**
  - Represents the physical implementation of the new system.

Guidelines for Drawing DFDs

- **Completeness**
  - DFD must include all components necessary for system.
  - Each component must be fully described in the project dictionary or CASE repository.
- **Consistency**
  - The extent to which information contained on one level of a set of nested DFDs is also included on other levels

Guidelines for Drawing DFDs (Cont.)

- **Timing**
  - Time is not represented well on DFDs.
  - Best to draw DFDs as if the system has never started and will never stop.
- **Iterative Development**
  - Analyst should expect to redraw diagram several times before reaching the closest approximation to the system being modeled.

Guidelines for Drawing DFDs (Cont.)

- **Primitive DFDs**
  - Lowest logical level of decomposition
  - Decision has to be made when to stop decomposition

Guidelines for Drawing DFDs (Cont.)

- Rules for stopping decomposition
  - When each process has been reduced to a single decision, calculation or database operation
  - When each data store represents data about a single entity

Guidelines for Drawing DFDs (Cont.)

- Rules for stopping decomposition, cont.
  - When the system user does not care to see any more detail
  - When every data flow does not need to be split further to show that data are handled in various ways

Guidelines for Drawing DFDs (Cont.)

- Rules for stopping decomposition, cont.
  - When you believe that you have shown each business form or transaction, online display and report as a single data flow
  - When you believe that there is a separate process for each choice on all lowest-level menu options

Using DFDs as Analysis Tools

- **Gap Analysis** is the process of discovering discrepancies between two or more sets of data flow diagrams or discrepancies within a single DFD.
- Inefficiencies in a system can often be identified through DFDs.

Using DFDs in BPR

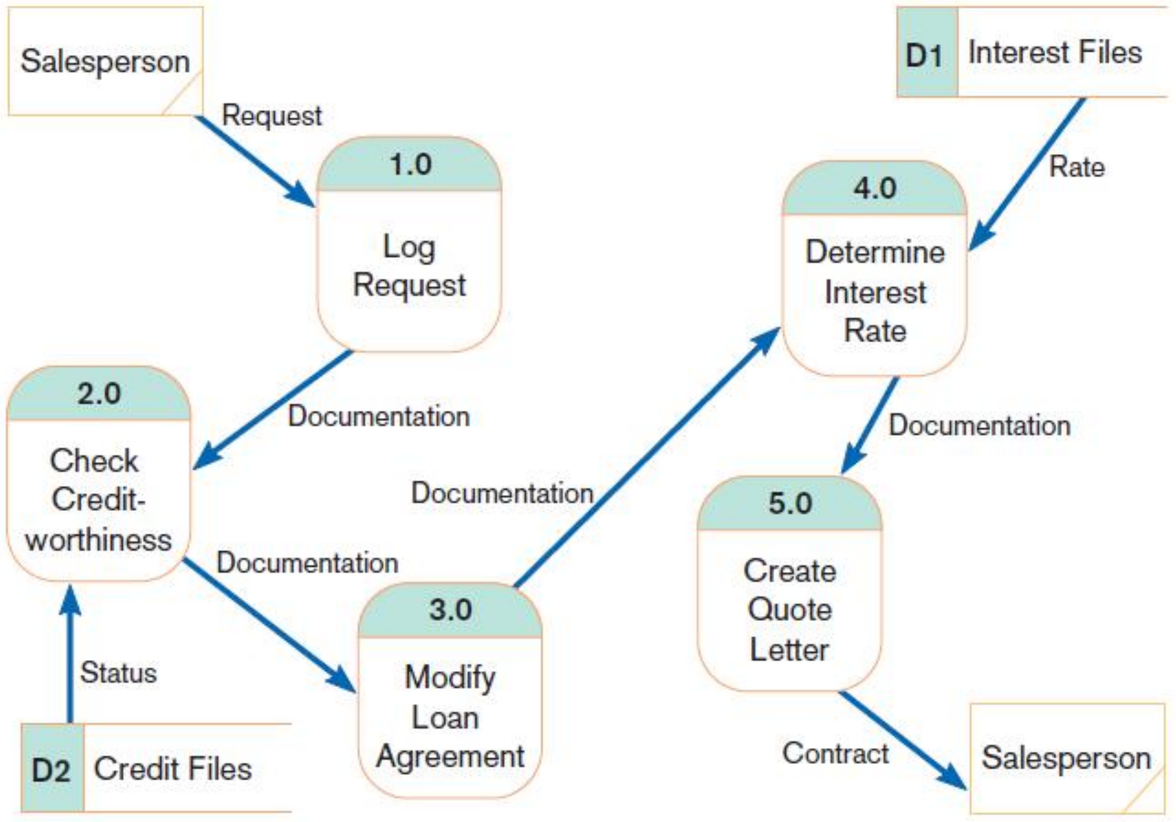


FIGURE 7-16
IBM Credit Corporation's primary work process before BPR
(*Source: Based on Hammer and Champy,* 1993.)

Using DFDs in BPR (Cont.)

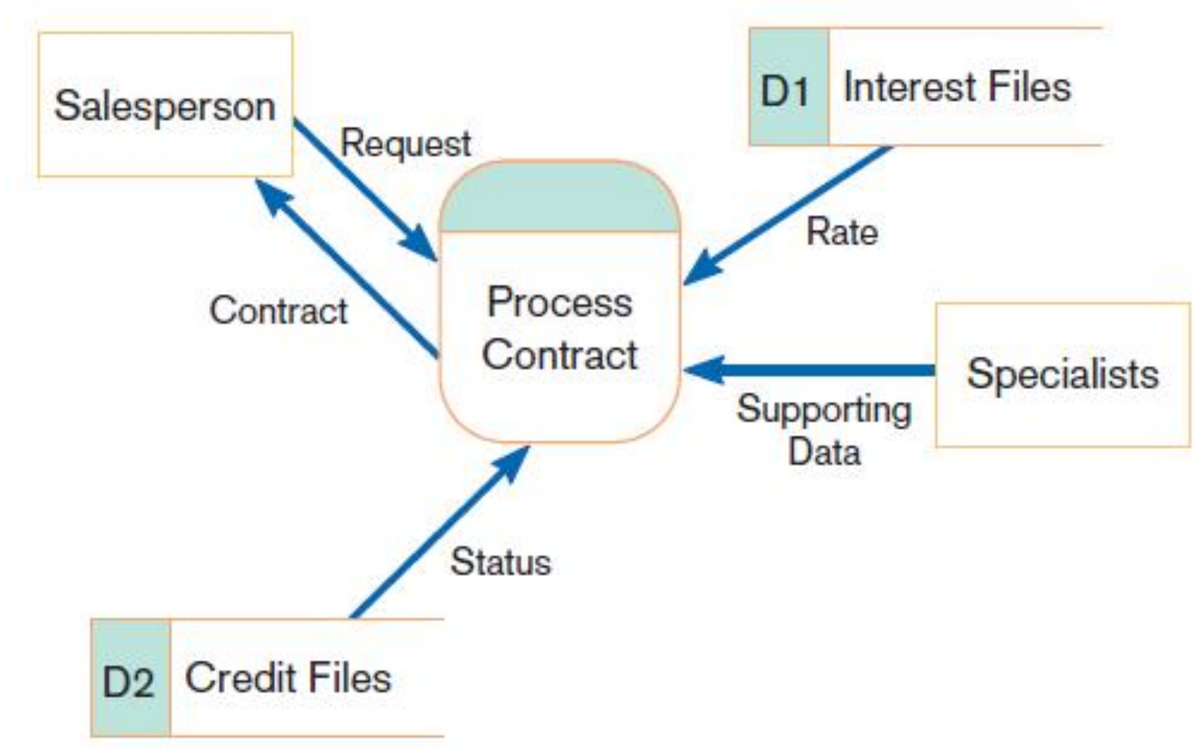


FIGURE 7-17
IBM Credit Corporation’s primary work process after BPR
(*Source: Based on Hammer and Champy,* 1993.)

Electronic Commerce Application: Process Modeling using Data Flow Diagrams

- Process modeling for Pine Valley Furniture's Webstore
  - Completed JAD session.
  - Began translating the Webstore system structure into data flow diagrams.
    - Identified six high-level processes.

Electronic Commerce Application: Process Modeling using Data Flow Diagrams (Cont.)

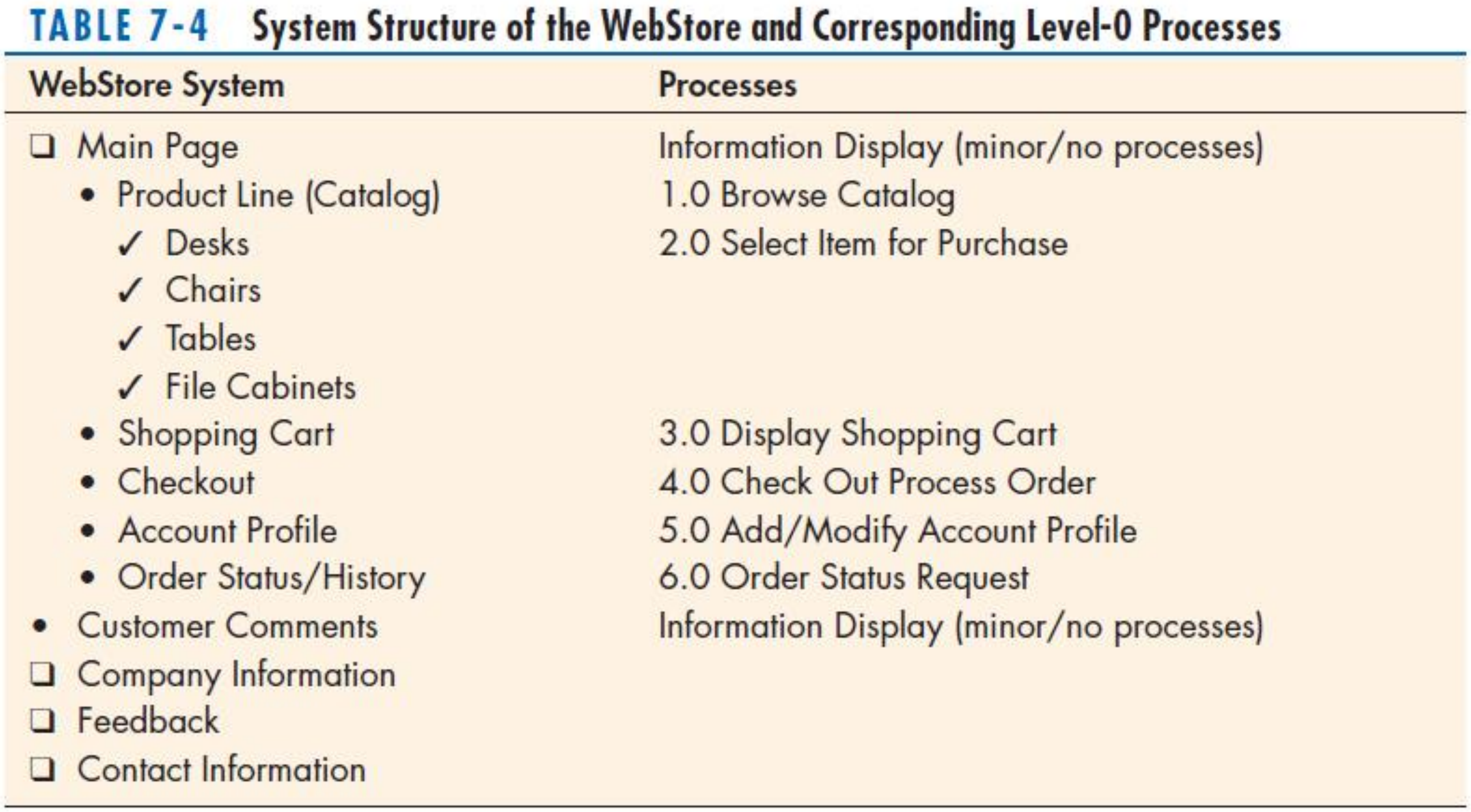


TABLE 7-4 System Structure of the WebStore and Corresponding Level-0 Processes

WebStore System	Processes
❑ Main Page	Information Display (minor/no processes)
• Product Line (Catalog)	1.0 Browse Catalog
✓ Desks	2.0 Select Item for Purchase
✓ Chairs	
✓ Tables	
✓ File Cabinets	
• Shopping Cart	3.0 Display Shopping Cart
• Checkout	4.0 Check Out Process Order
• Account Profile	5.0 Add/Modify Account Profile
• Order Status/History	6.0 Order Status Request
• Customer Comments	Information Display (minor/no processes)
❑ Company Information	
❑ Feedback	
❑ Contact Information	

Electronic Commerce Application: Process Modeling using Data Flow Diagrams

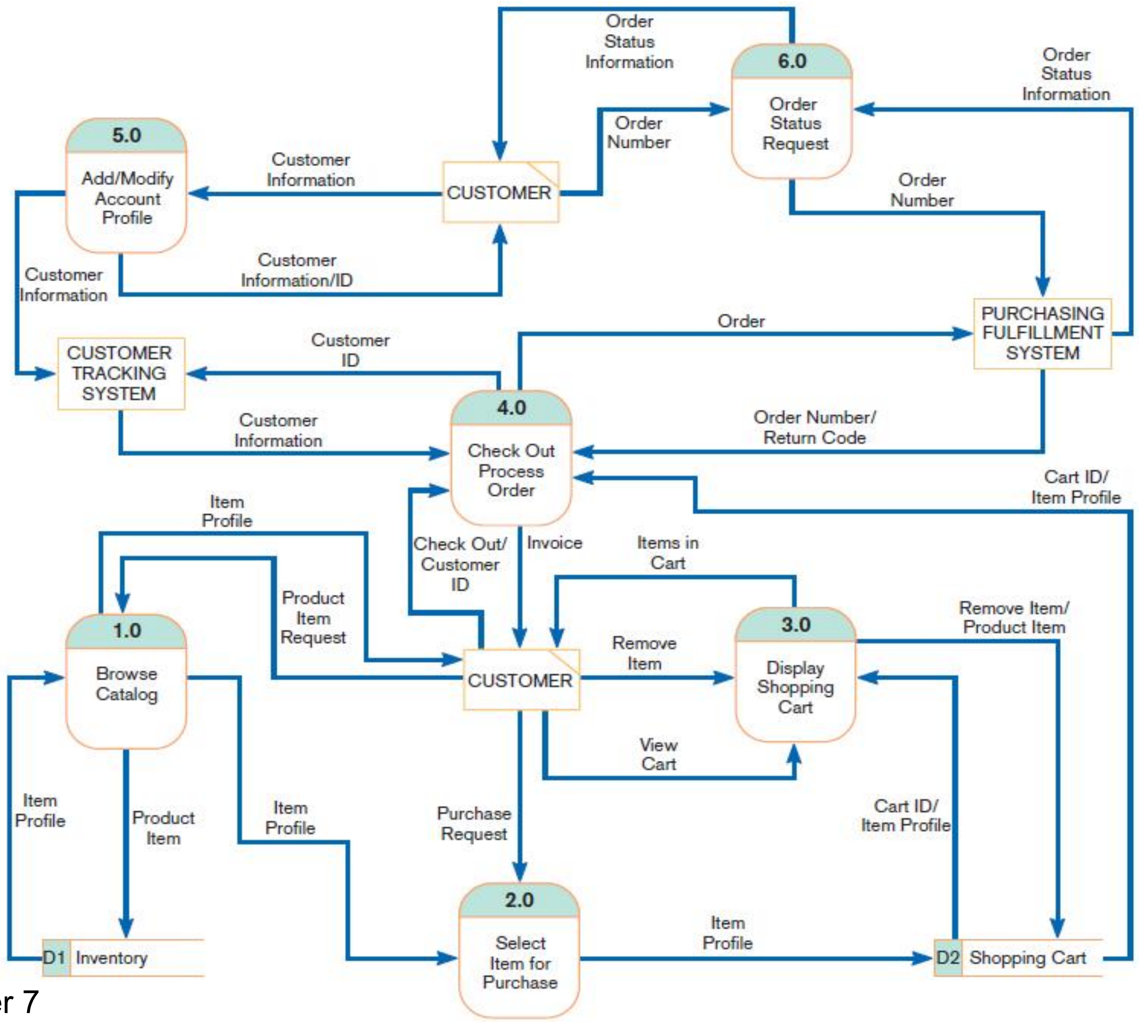


FIGURE 7-22
Level-0 data flow diagram for the WebStore