

# **Analysis**

## **Structuring System Requirements: Process Modeling**

Prepared by:

Hiranya Prasad Bastakoti

# Contents

- **Introduction**
- **Process Modeling**
- **Data Flow Diagrams**
- **Modeling Logic with Decision Tables,**
- **Decision Trees**
- **Pseudocodes**

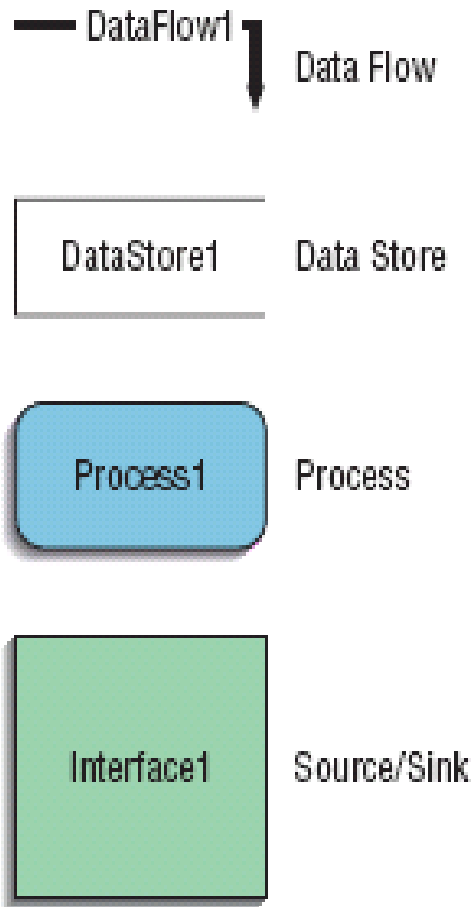
# Process Modeling

- Utilize information gathered during requirements determination
- Structure of the data is also modeled in addition to the processes
- Data-flow Diagrams (DFD)
  - Graphically represents the processes that capture, manipulate, store, and distribute data between a system and its environment and among system components

# Process Modeling

- Deliverables and Outcomes
  - Set of coherent, interrelated data-flow diagrams
  - Context data-flow diagram (DFD)
    - Scope of system
  - DFDs of current system
    - Enable analysts to understand current system
  - DFDs of new logical system
    - Technology independent
    - Show data flows, structure and functional requirements of new system
  - Documented in Project Dictionary or CASE Repository

# Data-Flow Diagramming Mechanics



**FIGURE 6-2**

Gane and Sarson identified four symbols to use in data-flow diagrams to represent the flow of data: data-flow symbol, data-store symbol, process symbol, and source/sink symbol. We use the Gane and Sarson symbols in this book.

# Data-Flow Diagramming Mechanics



- Data Flow
  - Depicts data that are in motion and moving as a unit in the system
  - Drawn as an arrow
  - Select a meaningful name to represent the aggregation of individual elements of data

# Data-Flow Diagramming Mechanics

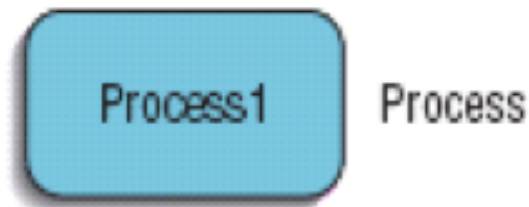
- Data Store

- Depicts data at rest
- May represent data in
  - File folder
  - Computer-based file
  - Notebook
- Drawn as a rectangle with the right vertical line missing
- Label includes name of the store as well as the number



# Data-Flow Diagramming Mechanics

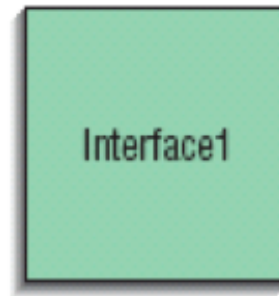
- Process
  - Depicts work or actions performed on data so that they are transformed, stored, or distributed
  - Drawn as a rectangle with rounded corners
  - Number of process as well as names are recorded



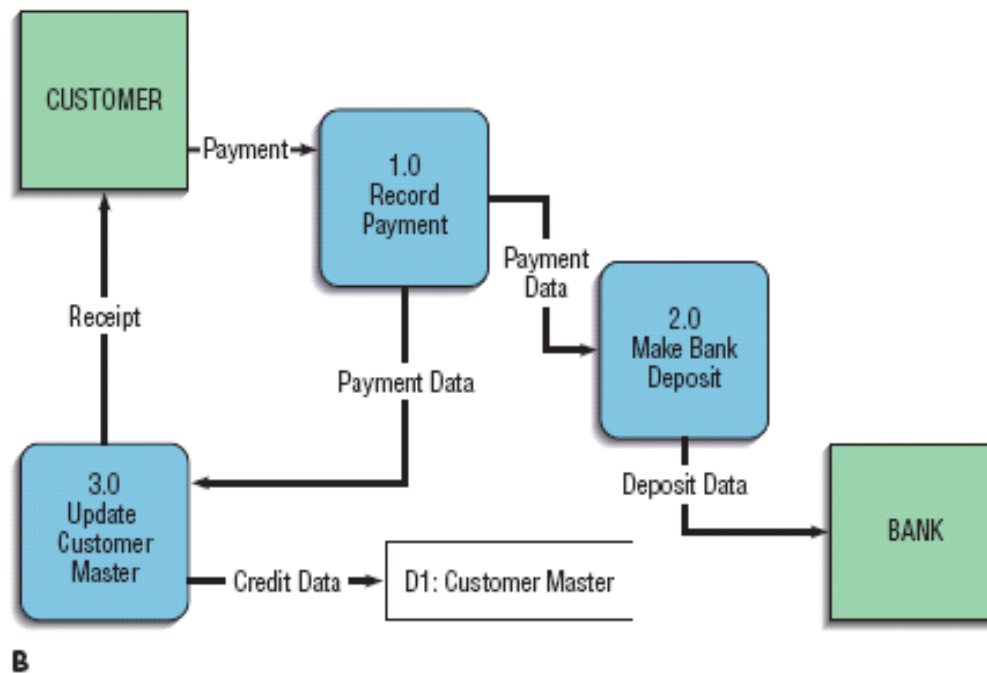
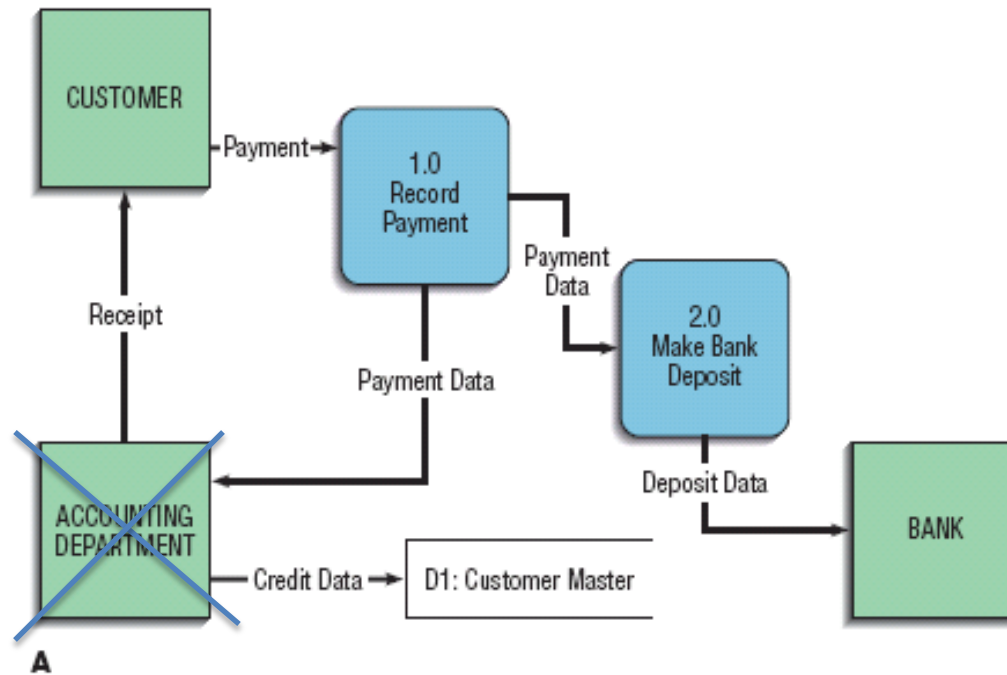


# Data-Flow Diagramming Mechanics

- Source/Sink
  - Depicts the origin and/or destination of data
  - Sometimes referred to as an external entity
  - Drawn as a square symbol
  - Name states what the external agent is
  - Because they are external, many characteristics are not of interest to us



Source/Sink



**FIGURE 6-3**  
**(A) An Incorrectly Drawn DFD Showing a Process as a Source/Sink**  
**(B) A DFD Showing Proper Use of a Process**

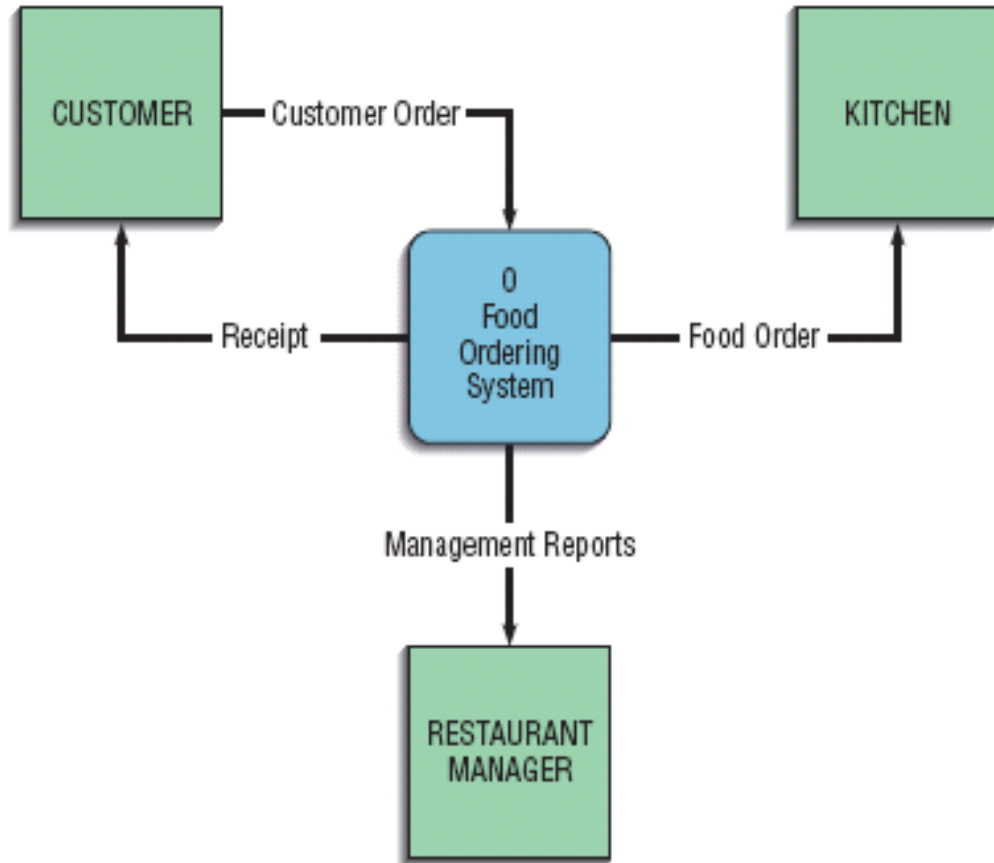
# Data-Flow Diagramming Definitions

- Context Diagram
  - A data-flow diagram of the scope of an organizational system that shows the system boundaries, external entities that interact with the system and the major information flows between the entities and the system
- Level-O Diagram
  - A data-flow diagram that represents a system's major processes, data flows, and data stores at a higher level

# Developing DFDs: An Example

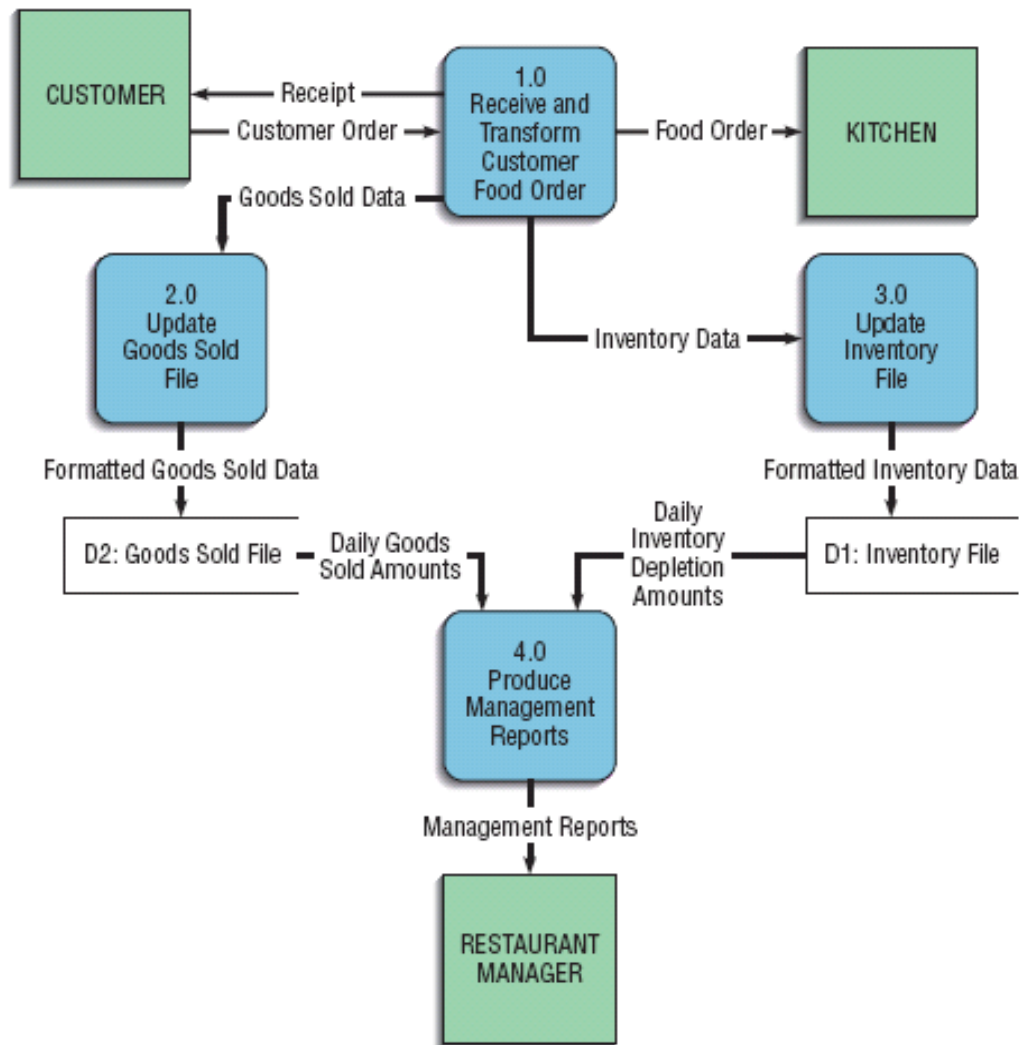
**FIGURE 6-4**  
**A Context Diagram**  
**of Hoosier Burger's**  
**Food-Ordering System**

The system includes one process (Food-Ordering System), four data flows (Customer Order, Receipt, Food Order, Management Reports), and three sources/sinks (Customer, Kitchen, and Restaurant Manager).



# Developing DFDs: An Example

- Next step is to expand the context diagram to show the breakdown of processes



**FIGURE 6-5**  
Four Separate Processes  
of the Hoosier Burger  
Food-Ordering System

# Data-Flow Diagramming Rules

- Basic rules that apply to all DFDs:
  - Inputs to a process are always different than outputs
  - Objects always have a unique name
    - In order to keep the diagram uncluttered, you can repeat data stores and data flows on a diagram

# Data-Flow Diagramming Rules

## ⦿ Process

- A. No process can have only outputs **(a miracle)**
- B. No process can have only inputs **(black hole)**
- C. A process has a verb phrase label

## ⦿ Data Store

- D. Data cannot be moved from one store to another
- E. Data cannot move from an outside source to a data store
- F. Data cannot move directly from a data store to a data sink
- G. Data store has a noun phrase label

## ⦿ Source/Sink

- H. Data cannot move directly from a source to a sink
- I. A source/sink has a noun phrase label



# Data-Flow Diagramming Rules

## ◎ Data Flow

- J. A data flow has only one direction of flow between symbols
- K. A fork means that exactly the same data go from a common location to two or more processes, data stores, or sources/sinks
- L. A join means that exactly the same data come from any 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
- N. A data flow to a data store means update
- O. A data flow from a data store means retrieve or use
- P. A data flow has a noun phrase label

# Decomposition of DFDs

- Functional Decomposition
  - Act of going from one single system to many component processes
  - Repetitive procedure
  - Lowest level is called a primitive DFD
- Level- $n$  Diagrams
  - A DFD that is the result of  $n$  nested decompositions of a series of subprocesses from a process on a level-0 diagram

# Balancing DFDs

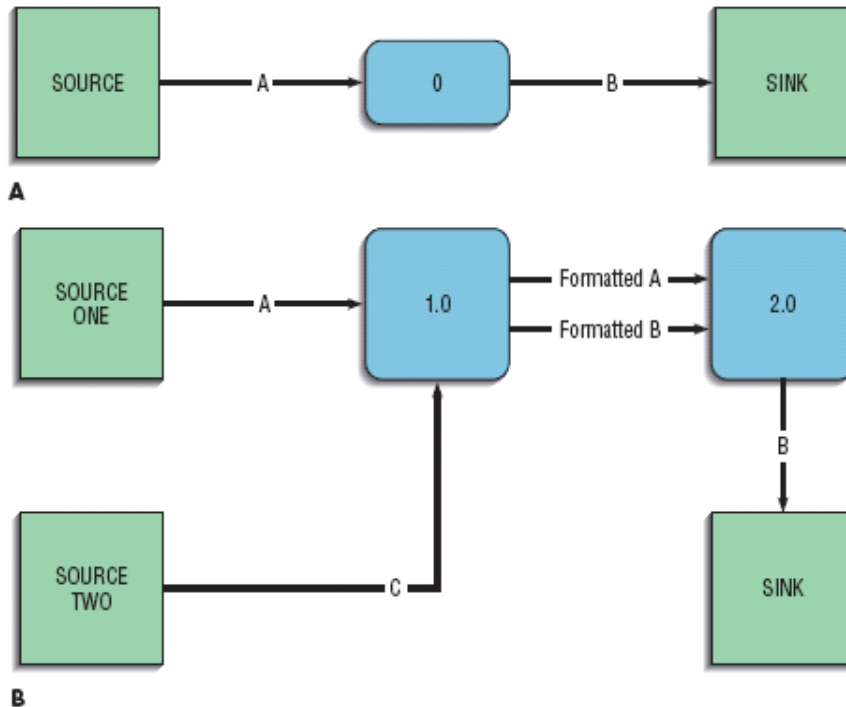
- When decomposing a DFD, you must conserve inputs to and outputs from a process at the next level of decomposition
  - This is called balancing
- Example: Hoosier Burgers
  - In Figure 6-4, one input to the system
    - Customer order
  - Three outputs:
    - Customer receipt
    - Food order
    - Management reports

# Balancing DFDs

- Example
  - Notice Figure 6-5. We have the same inputs and outputs
  - No new inputs or outputs have been introduced
  - We can say that the context diagram and level-0 DFD are balanced

# Balancing DFDs

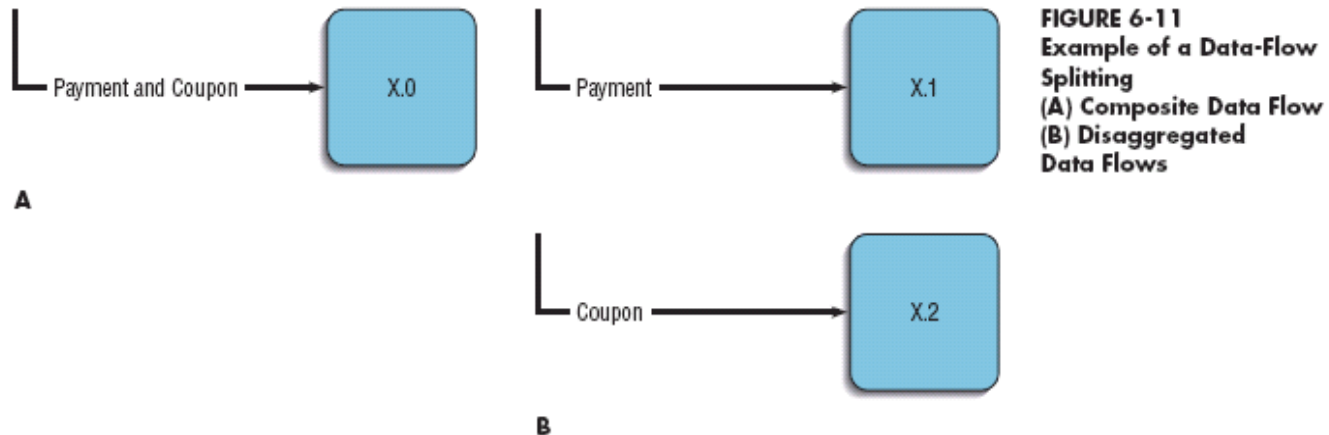
## An Unbalanced Example



- In context diagram, we have one input to the system, A and one output, B
- Level-0 diagram has one additional data flow, C
- These DFDs are not balanced

# Balancing DFDs

- We can split a data flow into separate data flows on a lower level diagram



# Balancing DFDs

## Four Additional Advanced Rules

**TABLE 6-3: 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 input 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, either to another process or to a data store outside the process or on 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 store 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 source/sink square, to indicate a repeated symbol.

Source: Adapted from J. Celko, "I. Data Flow Diagrams," *Computer Language* 4 (January 1987), 41–43.

# Guidelines for Drawing DFDs

## 1. Completeness

- DFD must include all components necessary for the system
- Each component must be fully described in the project dictionary or CASE repository

## 2. 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

## 3. Timing

- Time is not represented well on DFDs
- Best to draw DFDs as if the system has never started and will never stop

## 4. Iterative Development

- Analyst should expect to redraw diagram several times before reaching the closest approximation to the system being modeled

# Guidelines for Drawing DFDs

## 5. Primitive DFDs

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

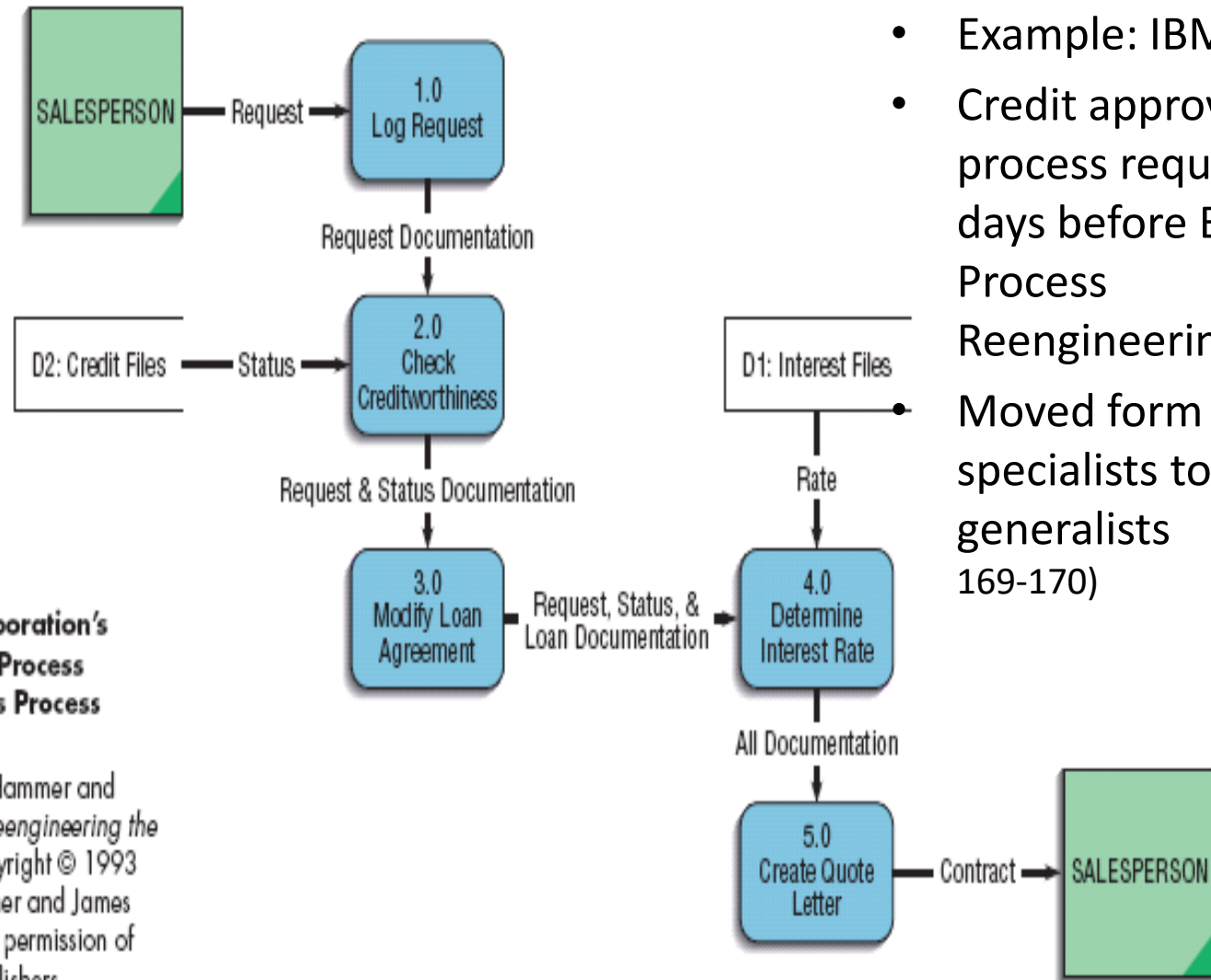
# Guidelines for Drawing DFDs

- 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 one entity
  - When the system user does not care to see more detail
  - When there is no need to split data flow further to show handling of different data elements
  - When you believe that you have shown each business form or transaction, online display, report as one 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
  - The process of discovering discrepancies between two or more sets of data-flow diagrams or discrepancies within a single DFD (see p.169)
- Inefficiencies in a system can often be identified through DFDs

# Using DFDs in Business Process



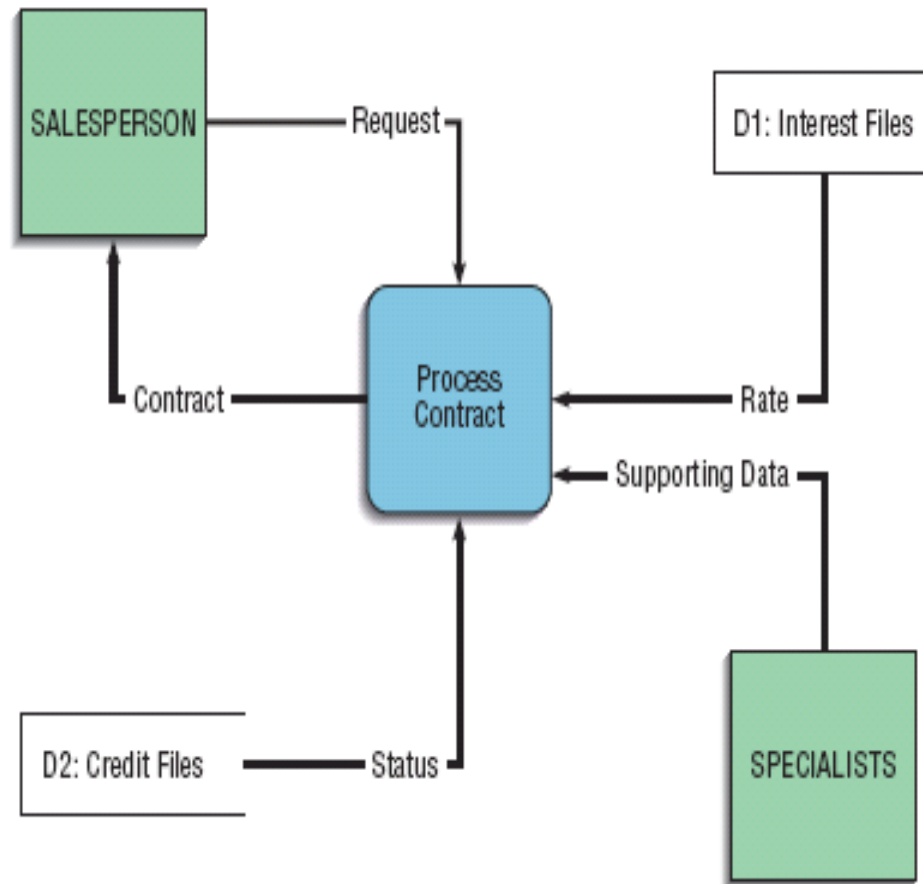
- Example: IBM Credit
- Credit approval process required six days before Business Process Reengineering Moved from specialists to generalists (see p. 169-170)

**FIGURE 6-12**  
**IBM Credit Corporation's**  
**Primary Work Process**  
**Before Business Process**  
**Reengineering**

Source: Michael Hammer and James Champy, *Reengineering the Corporation*. Copyright © 1993 by Michael Hammer and James Champy. Used by permission of HarperCollins Publishers.

# Using DFDs in Business Process Reengineering

- After Business Reprocess Engineering, IBM was able to process 100 times the number of transactions in the same amount of time



# Logic Modeling

- Data-flow diagrams do not show the logic inside the processes
- Logic modeling involves representing internal structure and functionality of processes depicted on a DFD
- Utilize:
  - Flow charts
  - Pseudo code
  - Decision Tables

# Modeling Logic with Decision Tables

- A matrix representation of the logic of a decision
- Specifies the possible conditions and the resulting actions
- Best used for complicated decision logic



# Modeling Logic with Decision Tables

- Consists of three parts:
  - Condition stubs
    - Lists condition relevant to decision
  - Action stubs
    - Actions that result for a given set of conditions
  - Rules
    - Specify which actions are to be followed for a given set of conditions

# Modeling Logic with Decision Tables

## ◎ Indifferent Condition

- › Condition whose value does not affect which action is taken for two or more rules

## ◎ Standard procedure for creating decision tables:

- › Name the conditions and values each condition can assume
- › Name all possible actions that can occur
- › List all possible rules
- › Define the actions for each rule (Figure 6-16)
- › Simplify the decision table (Figure 6-17)

Conditions/ Courses of Action	Rules											
	1	2	3	4	5	6	7	8	9	10	11	12
Type of item	P	N	P	N	P	N	P	N	P	N	P	N
Time of week	D	D	W	W	D	D	W	W	D	D	W	W
Season of year	A	A	A	A	S	S	S	S	H	H	H	H
Standing daily order	X				X				X			
Standing weekend order			X				X				X	
Minimum order quantity		X		X		X		X		X		X
Holiday reduction									X		X	
Summer reduction					X		X					

Type of item:

P = perishable

N = nonperishable

Time of week:

D = weekday

W = weekend

Season of year:

A = academic year

S = summer

H = holiday

Conditions/ Courses of Action	Rules						
	1	2	3	4	5	6	7
Type of item	P	P	P	P	P	P	N
Time of week	D	W	D	W	D	W	–
Season of year	A	A	S	S	H	H	–
Standing daily order	X		X		X		
Standing weekend order		X		X		X	
Minimum order quantity							X
Holiday reduction					X	X	
Summer reduction			X	X			

# Review Questions

1. Define: DFD, Structured English, Decision Table and Decision Tree.
2. List and Explain the features of DFD symbols
3. Draw a DFD upto level 2 of the followings:
  - University Registration system
  - Hotel Booking system
  - Patient Registration system
  - Online Shopping system
  - Library Management System

# **Reference Contents for DFDs**

- **DATA FLOW :**
- **A data flow** is a path for data to move from one part of the information system to another.
- A data flow in a DFD represents one or more data items. The data flow name appears above, below, or alongside the line.
- A data flow name consists of a singular noun and an adjective, if needed.
- Examples of data flow names are DEPOSIT, INVOICE PAYMENT, STUDENT GRADE, ORDER, and OMMISSION. Exceptions to the singular name rule are data flow names, such as GRADING PARAMETERS.
-

## DATA STORE :

- A **data store** is used in a DFD to represent data that the system Stores because one or more processes need to use the data at a later time.
- A data store name is a plural name consisting of a noun and adjectives, if needed.
- Examples of data store names are STUDENTS, ACCOUNTS RECEIVABLE, PRODUCTS, DAILY PAYMENTS, PURCHASE ORDERS, OUTSTANDING CHECKS, INSURANCE POLICIES, and EMPLOYEES



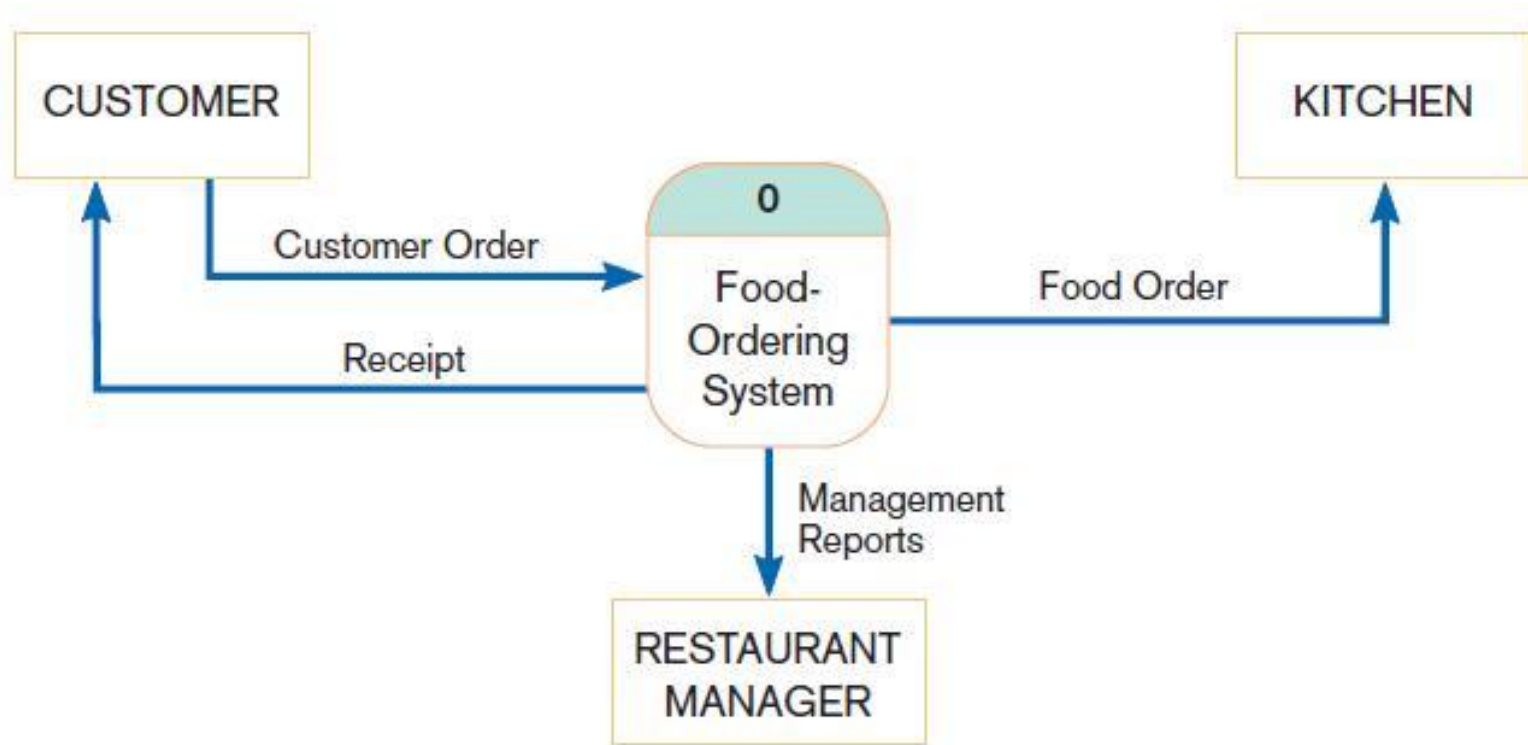
## ENTITY :

- The symbol for an **entity** is a rectangle, which may be shaded to make it look three-dimensional.
- The name of the entity appears inside the symbol. DFD entities also are called **terminators**, because they are data origins or final destinations.
- Systems analysts call an entity that supplies data to the system a **source**, and an entity that receives data from the system a **sink**.
- An entity name is the singular form of a department, outside organization, other information system, or person. An external entity can be a source or a sink or both, but each entity must be connected to a process by a data flow

# 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

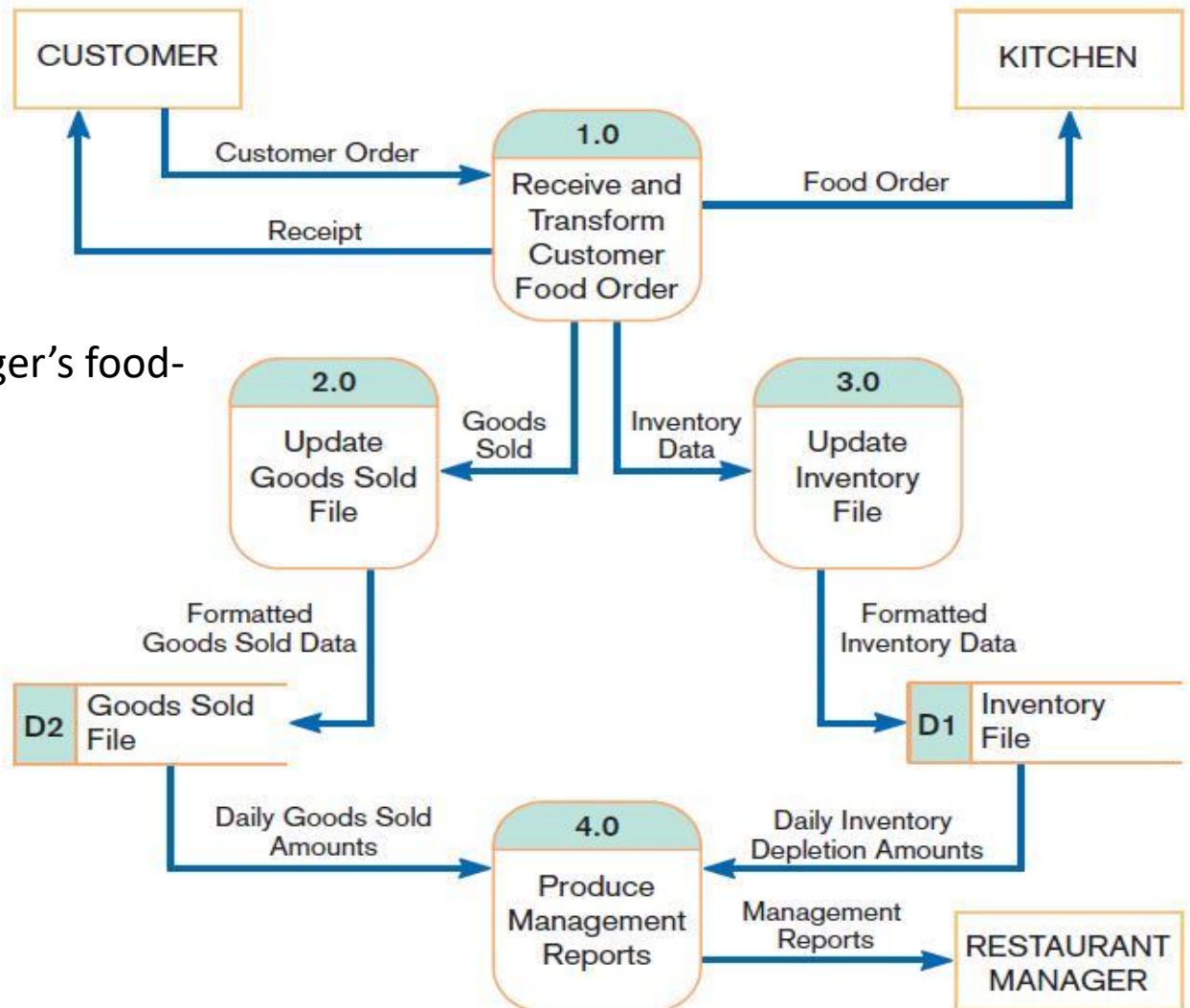


Context diagram of Hoosier Burger's food-ordering system

# Level-0 diagram

- **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



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.)

## **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.)

## Data Flow:

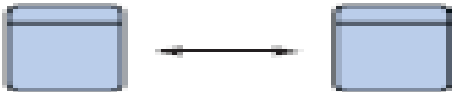





- J. A data flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated, however, by two separate arrows because these happen at different times.
- K. A fork in a data flow means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



# Correct and Incorrect uses of data flow

Correct and Incorrect Examples of Data Flows		
	Process to Process	✓
	Process to External Entity	✓
	Process to Data Store	✓
	External Entity to External Entity	✗
	External Entity to Data Store	✗
	Data Store to Data Store	✗

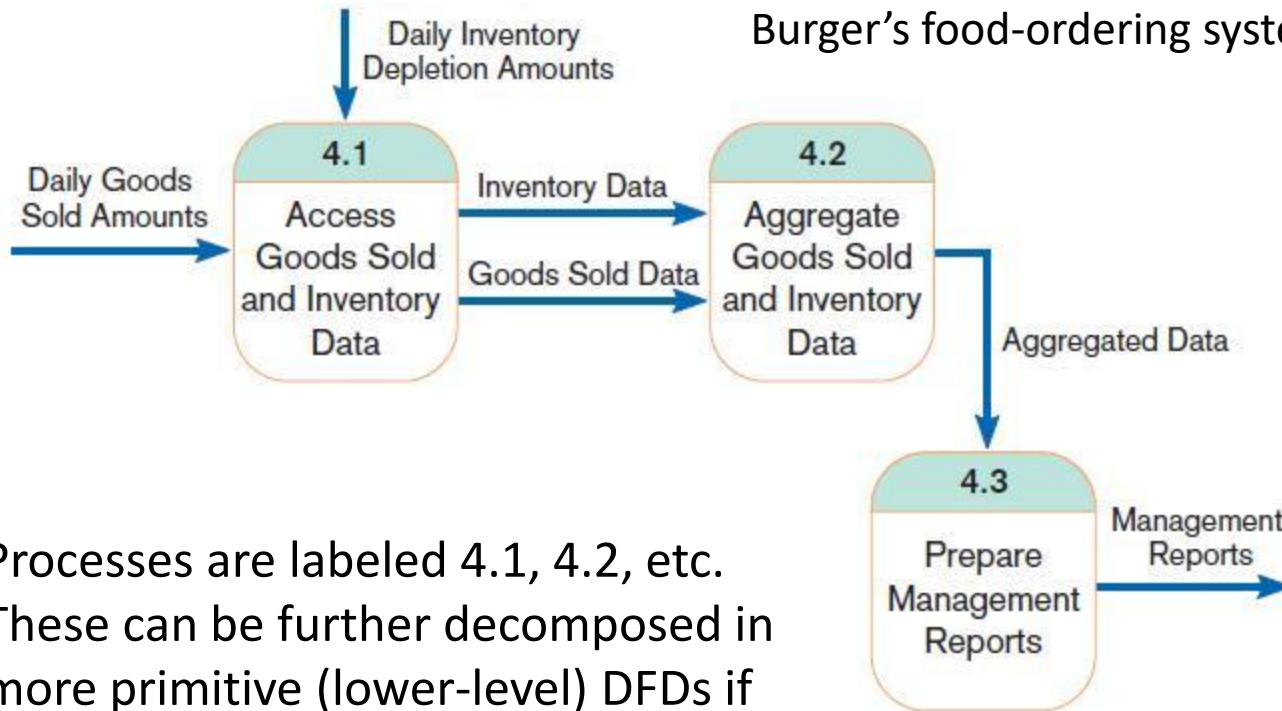
# Decomposition of DFDs

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

# 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



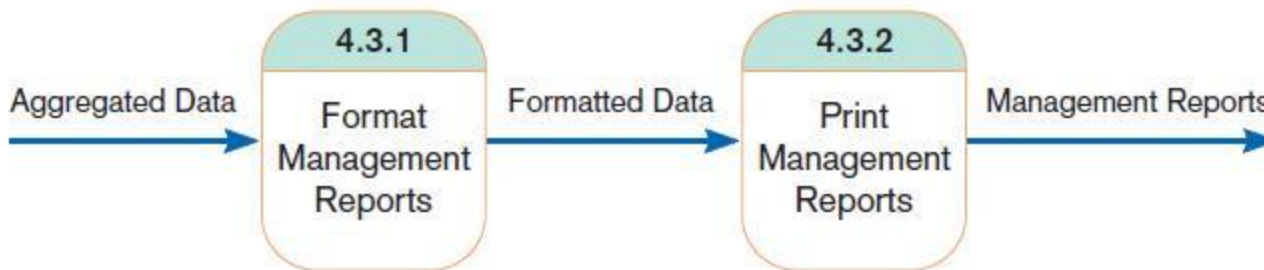
Level-1 DFD shows the sub-processes of one of the processes in the Level-0 DFD.

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

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

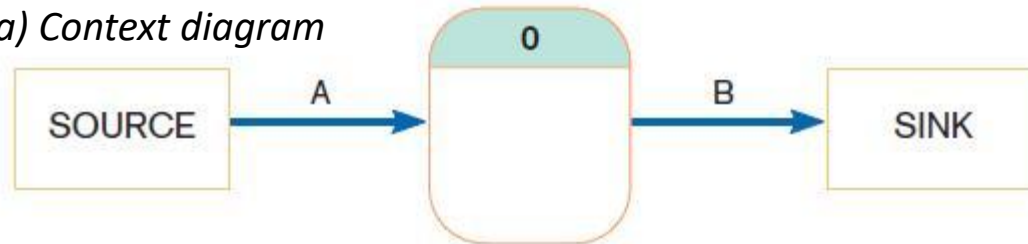
# Balancing DFDs (Cont.)

- 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.)

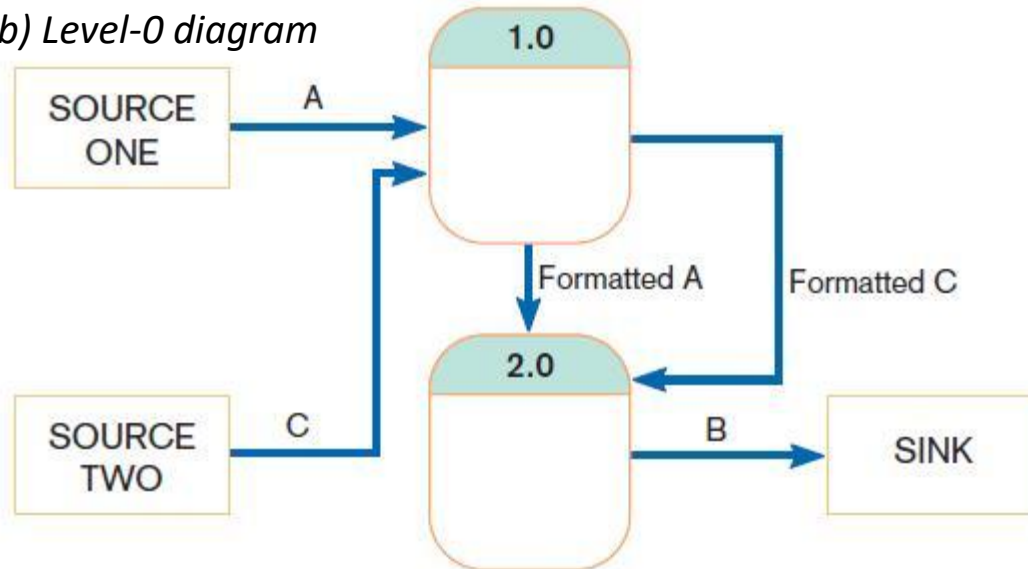
1 input  
1 output

(a) Context diagram



2 inputs  
1 output

(b) Level-0 diagram



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.)

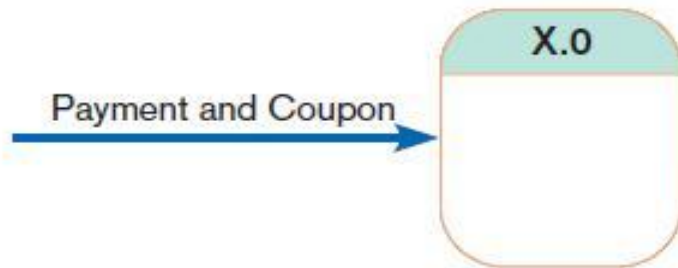
- **Data flow splitting** is when a composite data flow at a higher level is split and different parts go to different processes in the lower level DFD.
- The DFD remains balanced because the same data is involved, but split into two parts.



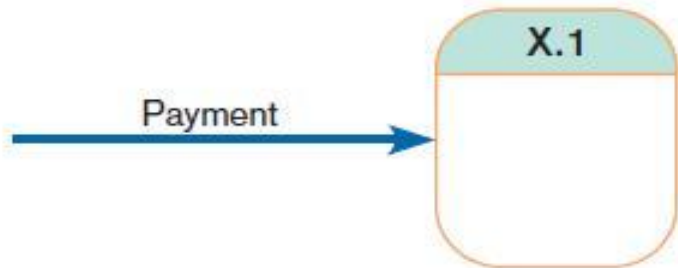
# Balancing DFDs (Cont.)

**FIGURE 7-11**

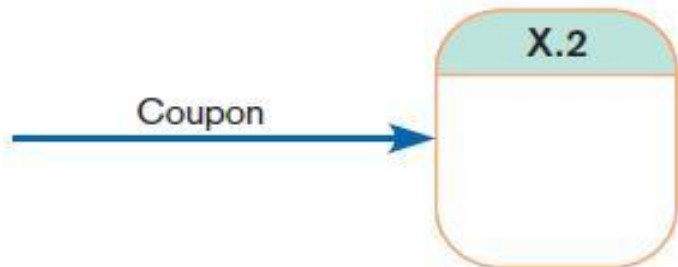
Example of data flow splitting



*(a) Composite data flow*



*(b) Disaggregated data flows*



# 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

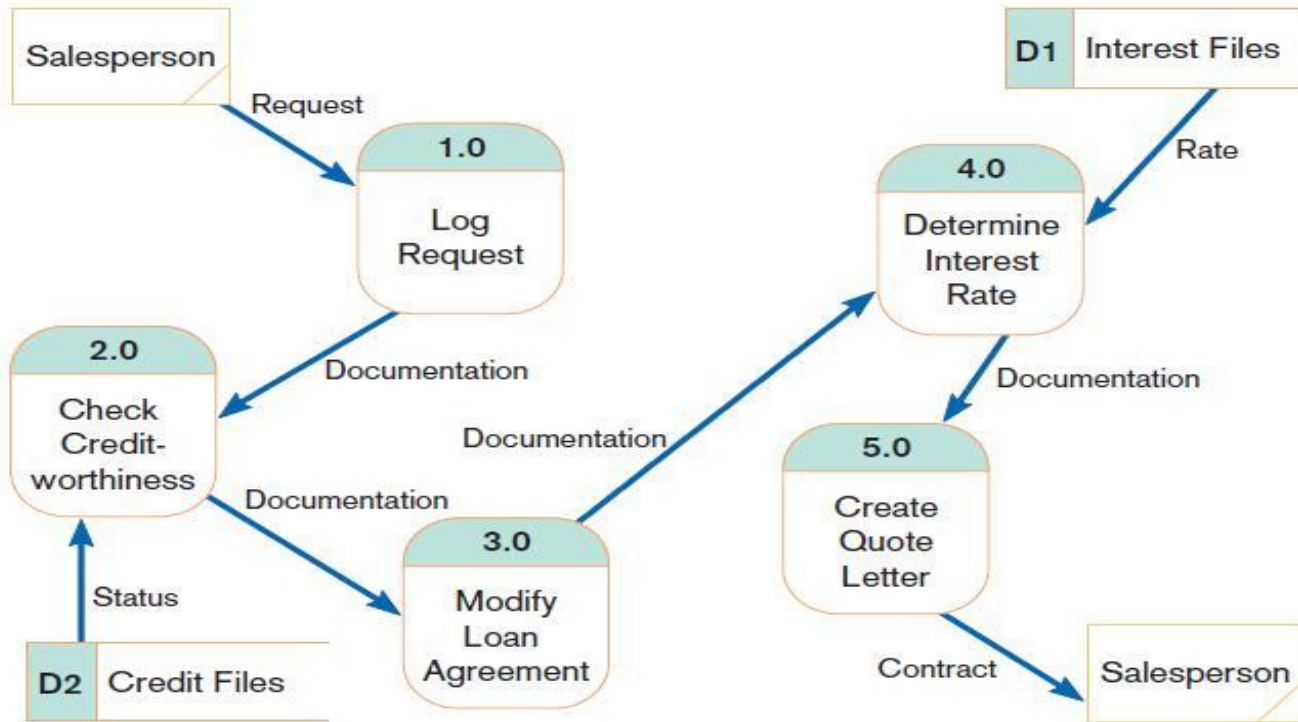
# Guidelines for Drawing DFDs (Cont.)

- 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

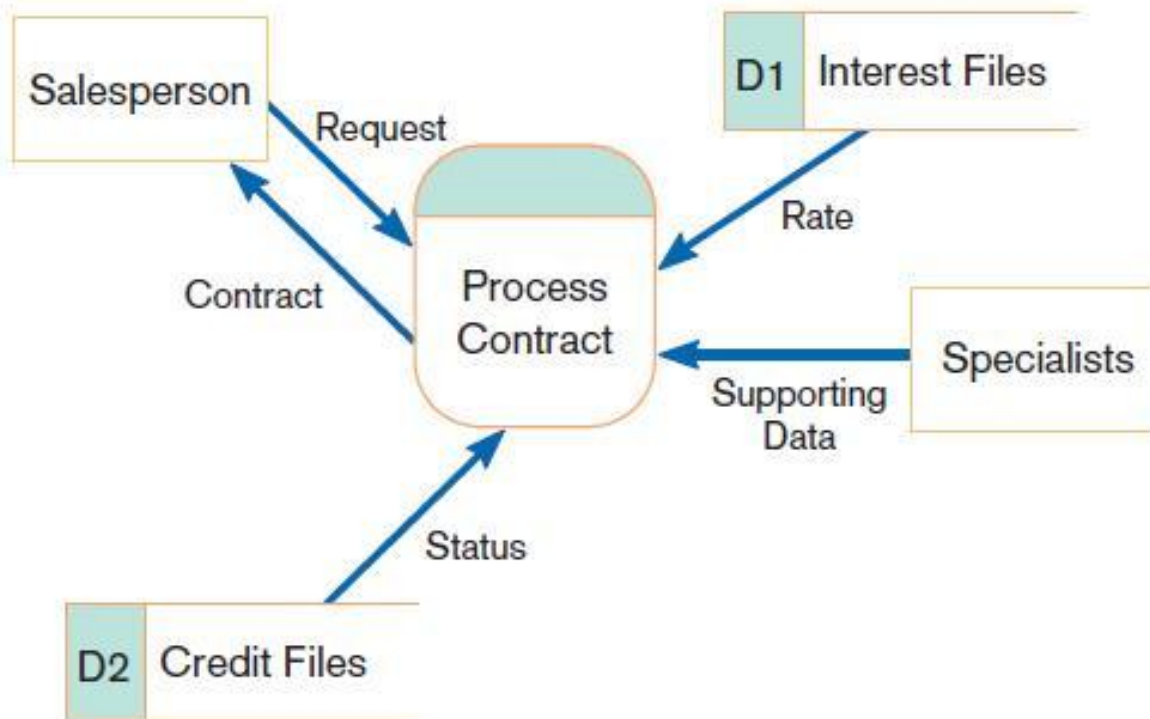
# Using DFDs in BPR



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



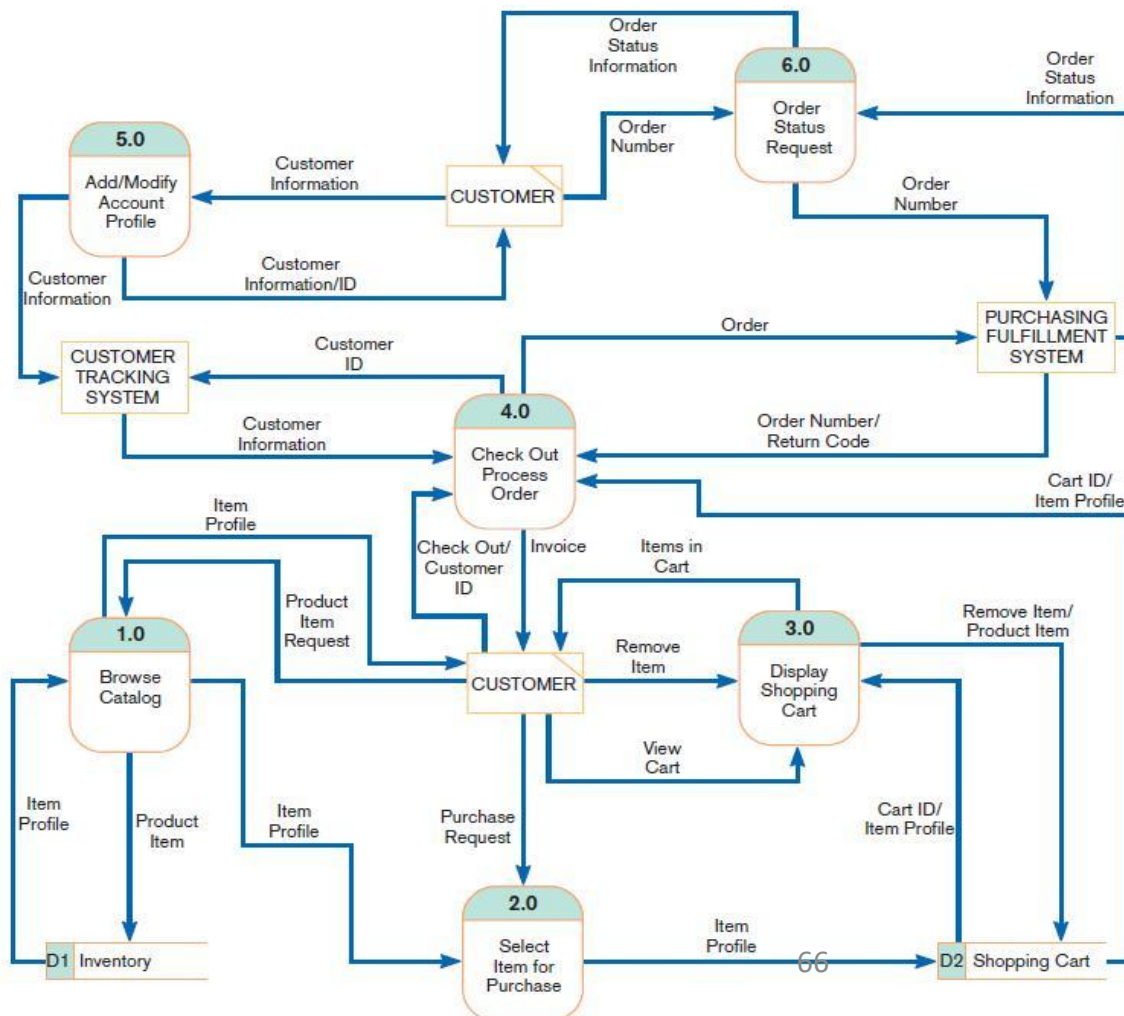
# Using DFDs in BPR (Cont.)



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

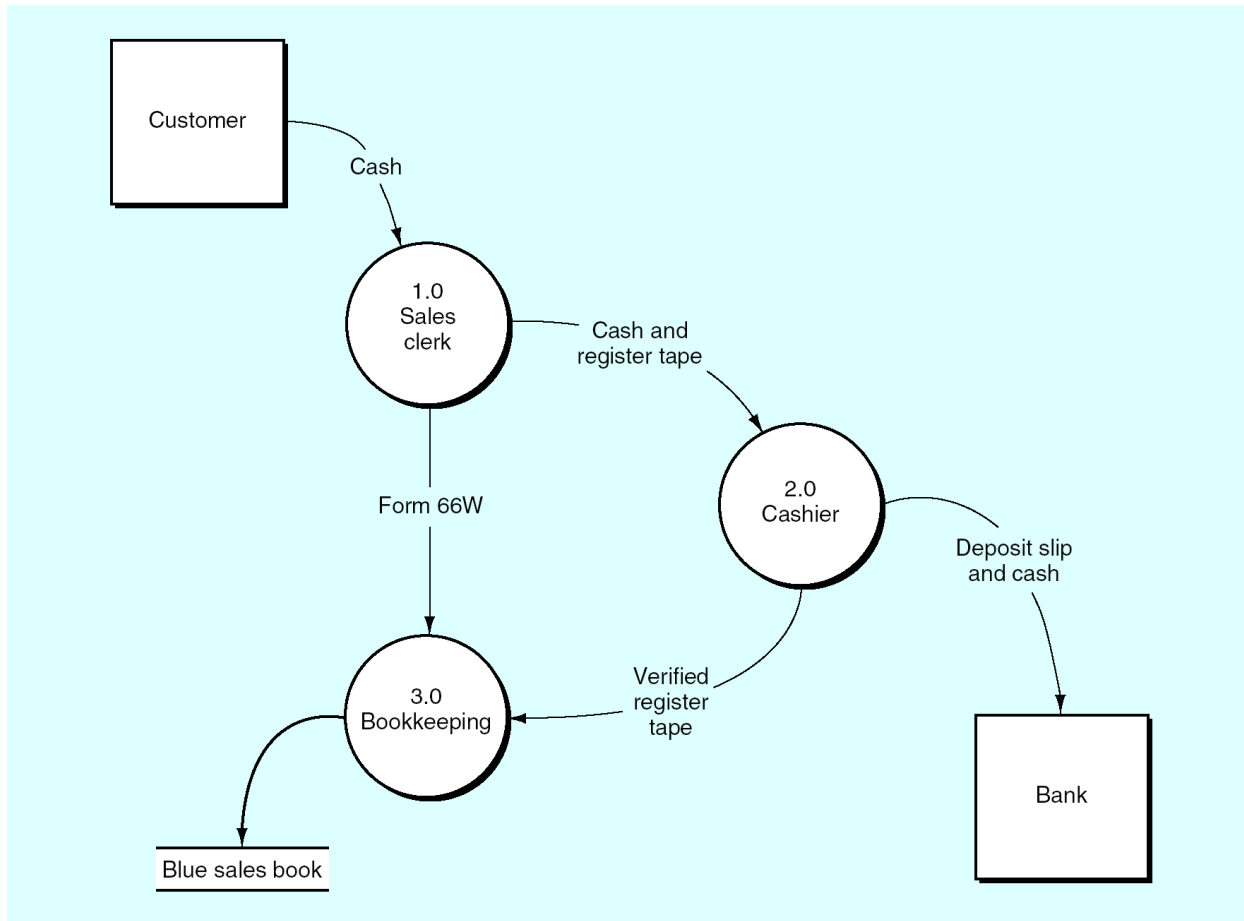
Level-0 data flow diagram for the WebStore

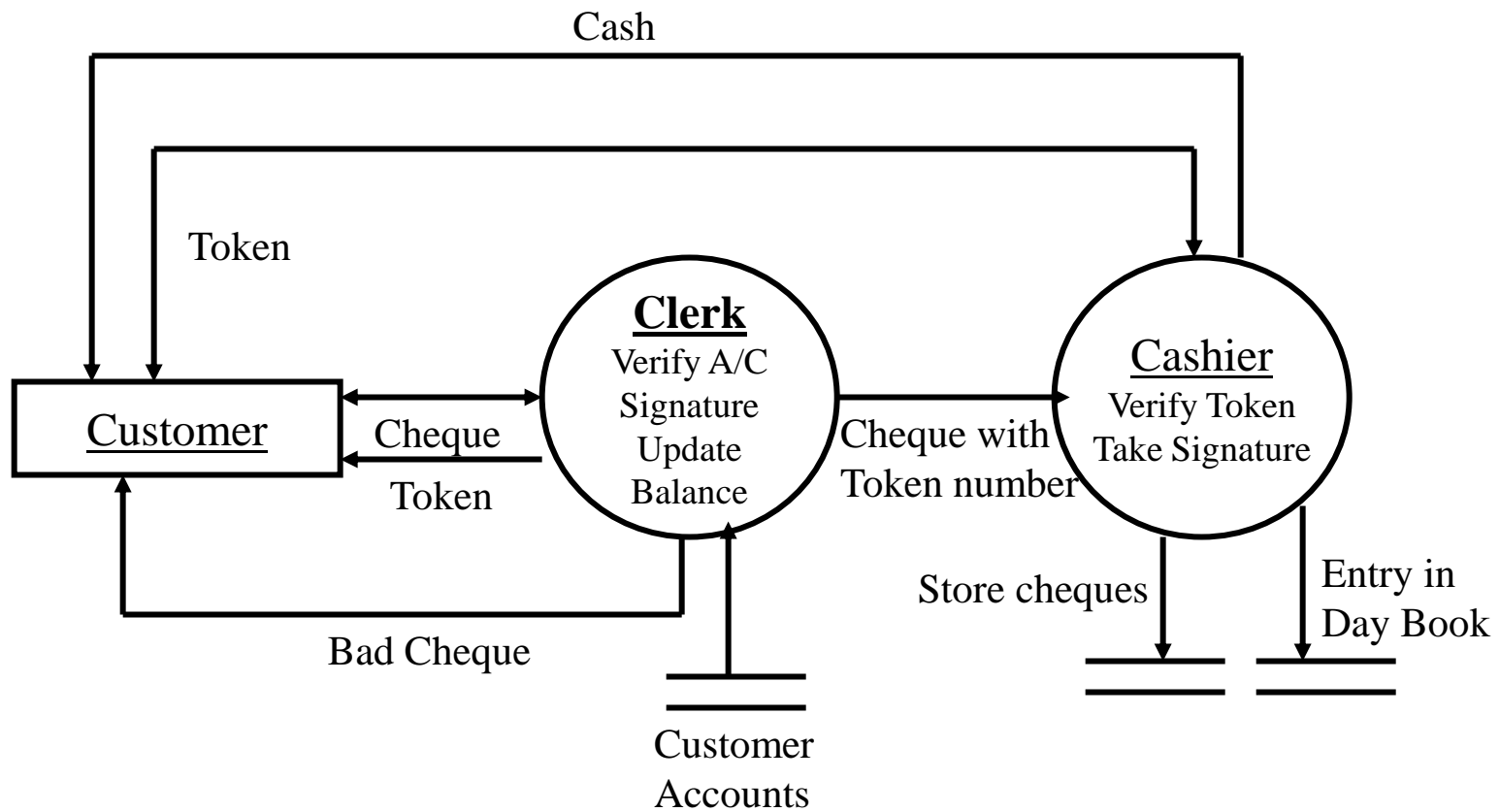


# Physical DFD

- A **physical data flow diagram** is a graphical representation of a system showing the system's internal and external entities, and the flows of data into and out of these entities.
  - A physical DFD specifies *where*, *how*, and by *whom* a system's processes are accomplished.
  - A physical DFD does not tell us *what* is being accomplished.
  - In the following slide, we see *where* the cash goes and *how* the cash receipts data are captured (that is, on the register tape), but we don't know exactly *what* was done by the sales clerk.

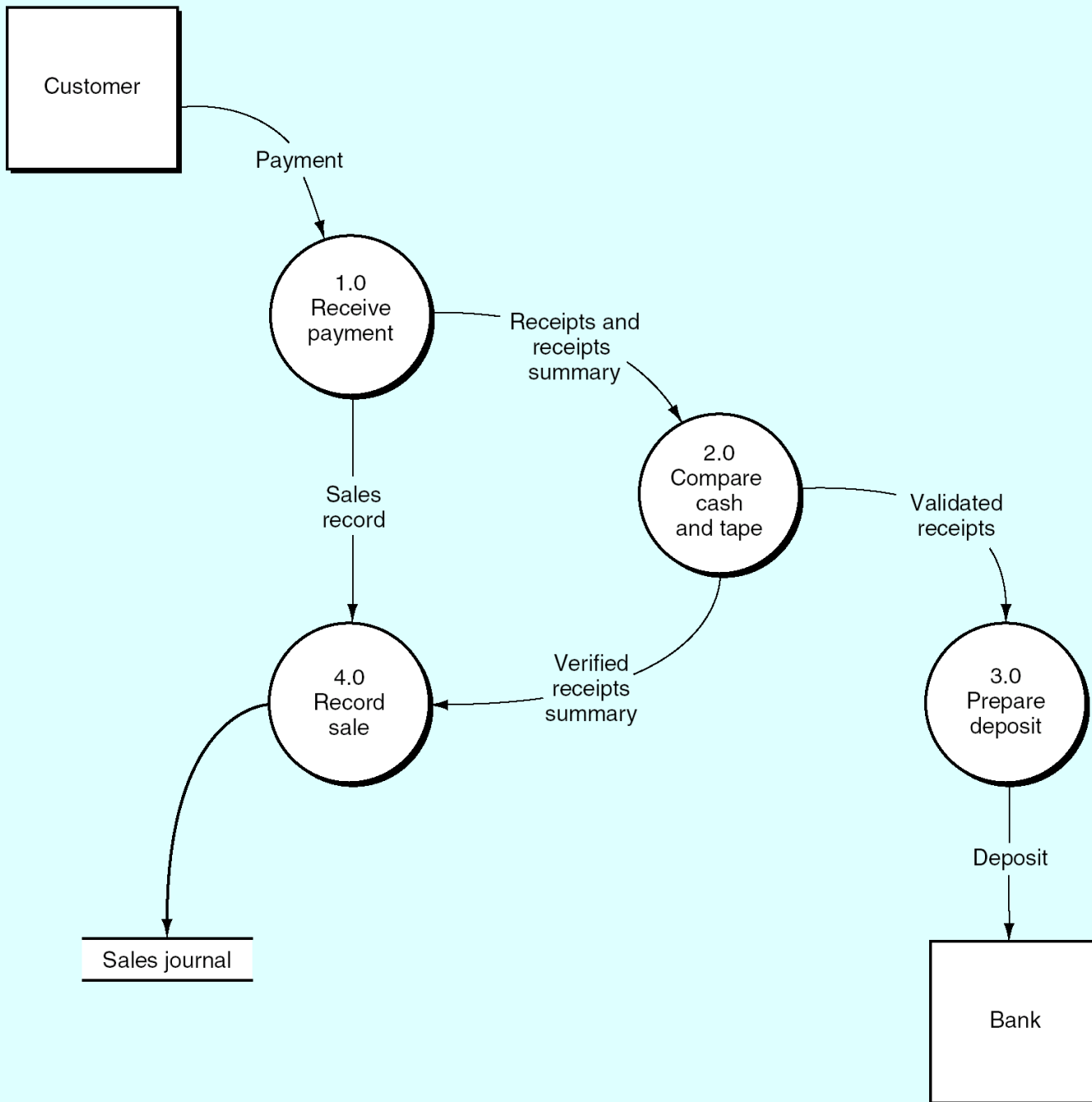
# Physical DFD





# Logical DFD

- A **logical data flow diagram** is a graphical representation of a system showing the system's processes (as bubbles), data stores, and the flows of data into and out of the processes and data stores.
  - We use a logical DFD to document information systems because we can represent the logical nature of a system—*what* tasks the system is doing— without having to specify *how*, *where*, or by *whom* the tasks are accomplished.
  - The advantage of a *logical* DFD (versus a *physical* DFD) is that we can concentrate on the functions that a system performs.
  - So, a logical DFD portrays a system's activities, whereas a physical DFD depicts a system's infrastructure.
  - We need both pictures to understand a system completely.



# Logical DFD

