King Saud University

College of Engineering

IE – 462: "Industrial Information Systems"

Spring – 2024 (2nd Sem. 1445H)

Chapter 4:

Structured Analysis and Functional Architecture Design – p2 – DFD – ii - Diagrams

Prepared by: Ahmed M. El-Sherbeeny, PhD

Lesson Overview

- Modeling IIS – (p1)
- Integrated Computer-Aided Manufacturing Definition 0 (IDEF0) – (p1)
- Data Flow Diagram (DFD) – (p2)
  i. Fundamentals
  ii. Diagramming Rules
  iii. Case Studies

DFD – part ii – Diagramming Rules

- Context Diagram
- Level-0 DFD
- Data Flow Diagramming Rules
- Decomposition of DFDs
- Food Ordering System Summary
- Balancing DFDs
- Guidelines for Drawing DFDs

Functional/Process Modeling:

2. Data Flow Diagram (DFD) – cont'd

Context Diagram

Functional Decomposition

- Functional decomposition: an iterative process of breaking a system description down into finer and finer details:
  - Creates a set of *hierarchically related* charts in which one process on a given chart is explained in greater detail on another chart
  - Continues until no sub-process can logically be broken down any further
  - Lowest level of a DFD is called a *primitive DFD*

TABLE 7-1 Deliverables for Process Modeling

1. Context DFD
2. DFDs of the system (adequately decomposed)
3. Thorough descriptions of each DFD component

Developing DFDs: Context Diagram

Context Diagram

- DFDs follow the same principles of hierarchic decomposition as IDEF0; highest level diagram is called the context diagram

- Context diagram includes *overall process* (i.e. scope of the system) and *sources and sinks* that interact with the overall process, showing:
  - system boundaries
  - external entities that interact with the system (sources/sinks)
  - *major* information flows between the entities and the system

Developing DFDs: Context Diagram (cont.)

Context Diagram (cont.)

- Context diagram consists of only *one process* symbol (labeled *0*), and *no data stores* are shown
- It is decomposed into a first-level diagram (level-0 diagram) that shows more details of the process and data flow

Developing DFDs: Context Diagram (cont.)

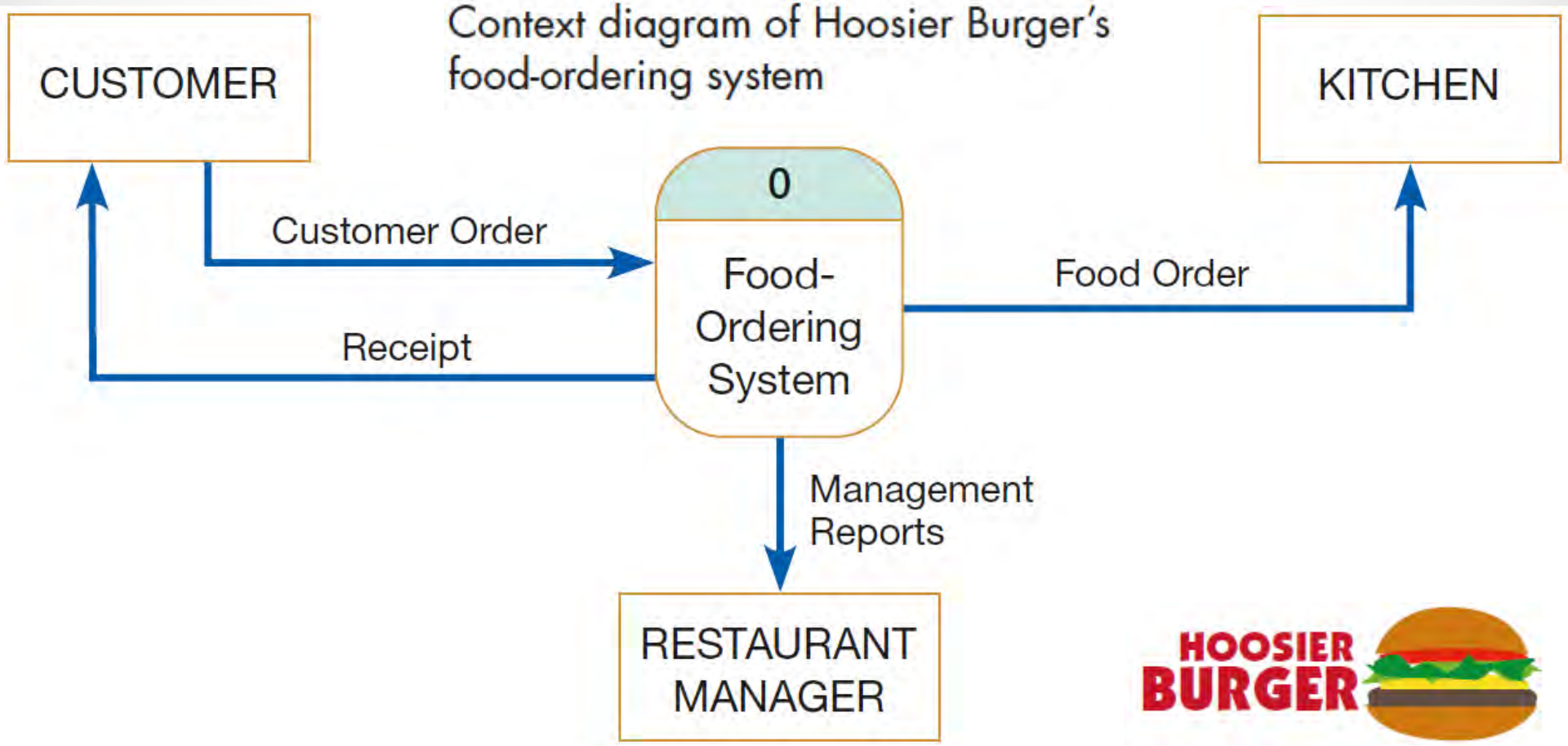
Context Diagram (cont.)

- Boundary relationships of context level are maintained at *each successive level* of decomposition
- Therefore, data flows from sources and to sinks that appear at the context level also appear at the first level of decomposition (this is called *Level-0 DFD*)
- Decomposition explores greater levels of detail, so data stores not represented at the context level may be introduced (e.g. filing cabinet, local to sub-system)
- In general, decomposition should be carried out to the degree necessary for the analyst to *understand the details* of the functions and data flows

Developing DFDs: Context Diagram (cont.)

Context Diagram (cont.)

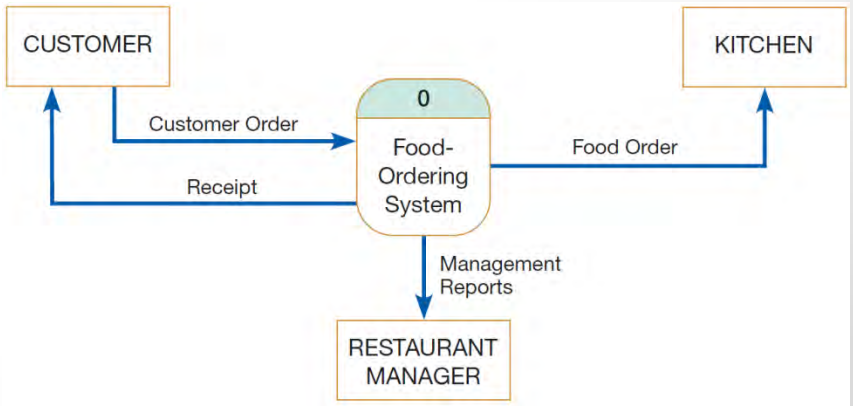
- e.g. see below context diagram for food ordering system



Developing DFDs: Context Diagram (cont.)

Context Diagram (cont.)

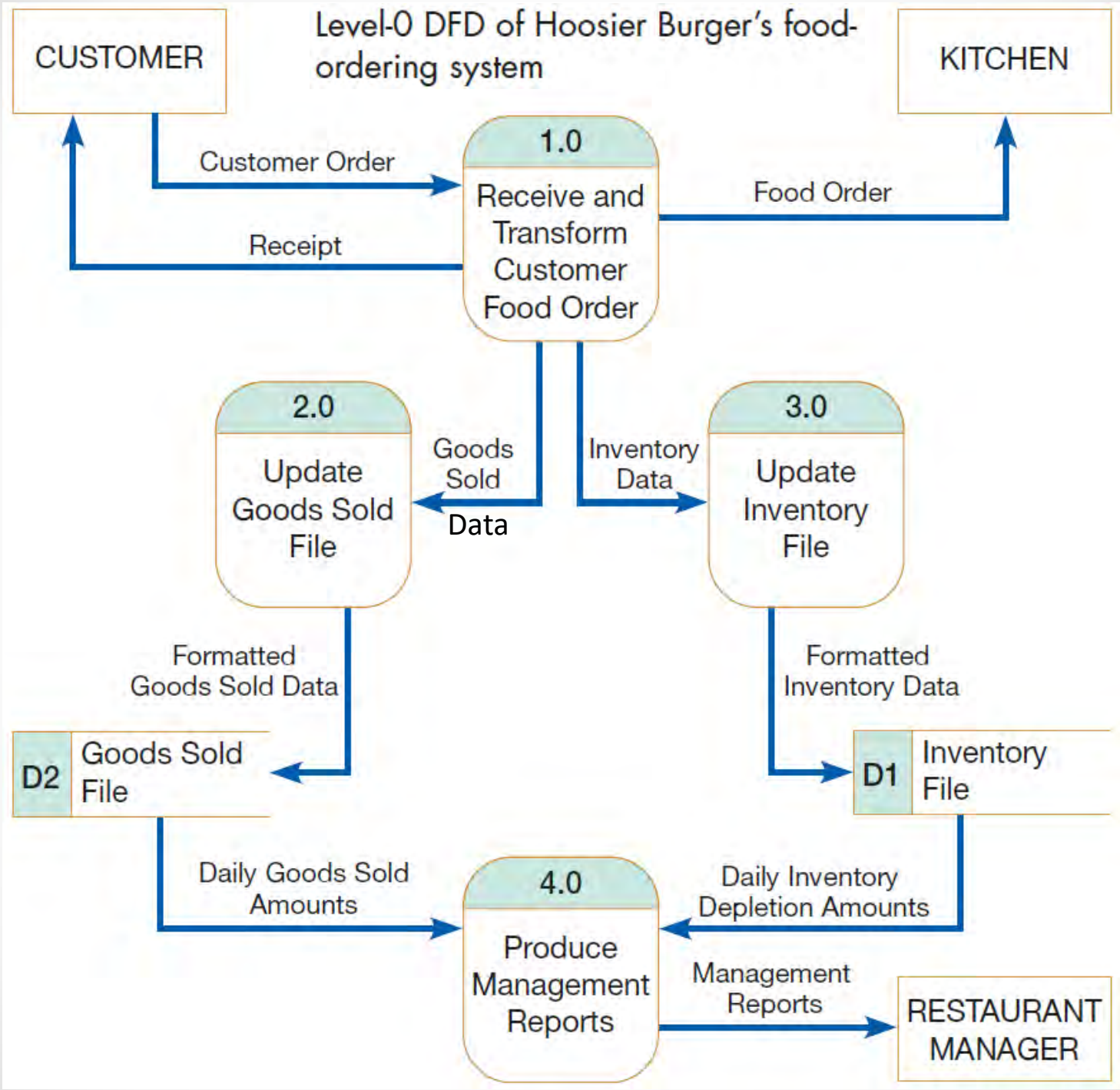
- Restaurant uses an IS to enable it to:
  - takes customer orders
  - send orders to the kitchen
  - monitor goods sold and inventory, and
  - generate reports for management
- Note how context diagram contains:
  - only *one* process (labeled 0, representing entire system)
  - *no* data stores (don't appear on context diagram)
  - four data flows, and
  - three sources/sinks (environmental boundaries)



Level-0 DFD

Developing DFDs: Level 0

- Level-0 diagram: 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
- see (following slide) of Level-0 diagram for food ordering system example

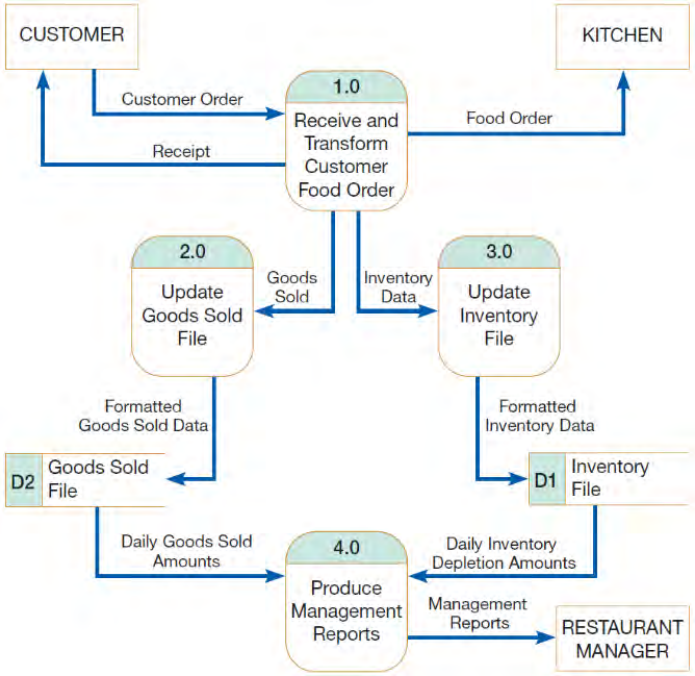

Level-0 DFD of Hoosier Burger's food-ordering system
CUSTOMER
KITCHEN
1.0
Receive and Transform Customer Food Order
Customer Order
Receipt
Food Order
2.0
Update Goods Sold File
3.0
Update Inventory File
Goods Sold Data
Inventory Data
Formatted Goods Sold Data
Formatted Inventory Data
D2
Goods Sold File
D1
Inventory File
Daily Goods Sold Amounts
Daily Inventory Depletion Amounts
4.0
Produce Management Reports
Management Reports
RESTAURANT MANAGER

Developing DFDs: Context Diagram (cont.)

Level-0 diagram (cont.)

- Result of process 1.0 (process customer order):
  1. food order is transmitted to the *kitchen*
  2. order is transformed into a list of *goods sold*
  3. order is transformed into *inventory data*
  4. process generates a *receipt* for the customer

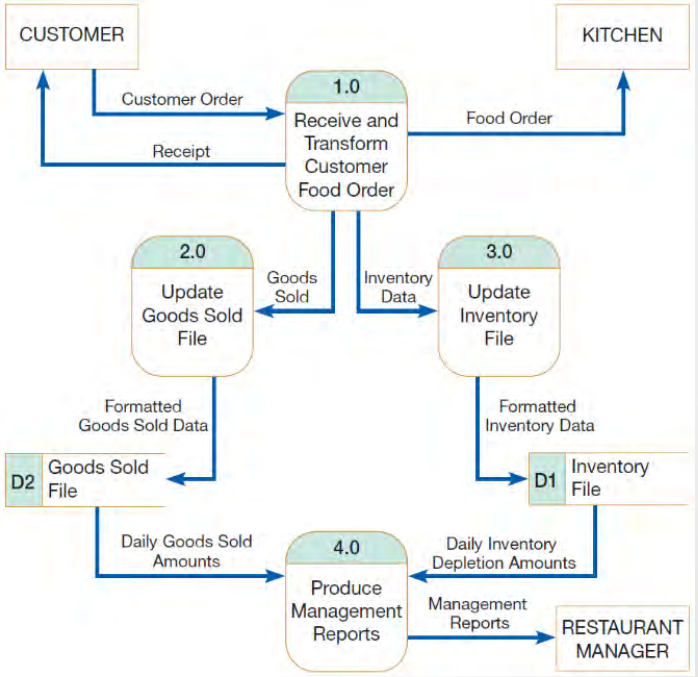
- Notice:
  - sources/sinks are the *same* in context diagram & level-0 diagram
  - each process in level-0 diagram has a number that ends in *.0* (corresponding to the *level number* of the DFD)
  - no need to worry about data flows to sinks (go to 'black box')



Developing DFDs: Context Diagram (cont.)

Level-0 diagram (cont.)

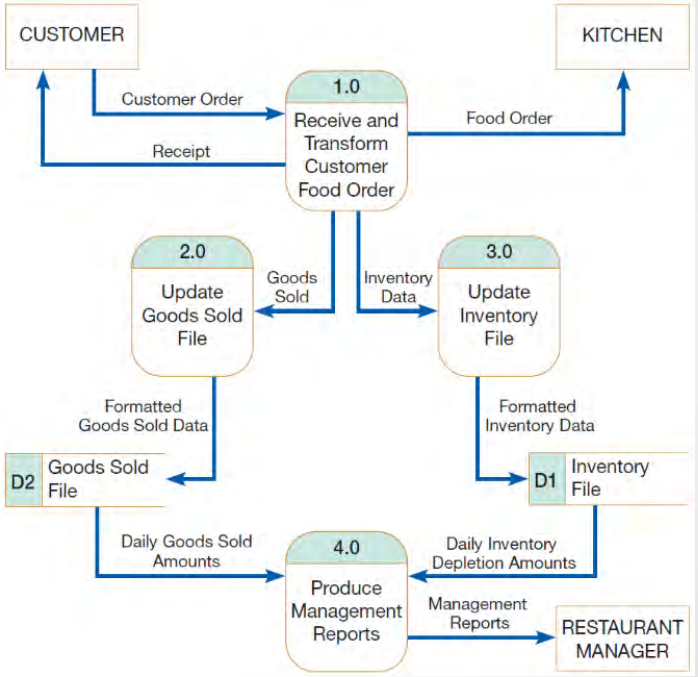
- Data flow from Process 1.0 to Process 3.0:
  - DFD hides many physical characteristics of the system it describes (e.g. *timing* of when data flow is produced, *how frequently* it is produced, or what *volume of data* is sent)
  - however, we know that Process 1.0 does provide this data to Process 3.0
  - also, Process 1.0 is coupled to Process 3.0 (i.e. Process 3.0 must *always* accept Inventory Data flow from Process 1.0)



Developing DFDs: Context Diagram (cont.)

Level-0 diagram (cont.)

- Data flow between Process 2.0 and Process 4.0:
  - output from Process 2.0 is placed in a data store and, later, when Process 4.0 needs such data, it reads *Daily Goods Sold Amounts* from this data store
  - Thus, Processes 2.0 and 4.0 are decoupled by placing a *buffer*, a data store, between them (i.e. each works at its own pace)



Data Flow Diagramming Rules

Data Flow Diagramming Rules

Process:

A. No process can have only outputs. It would be 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.

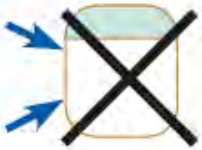
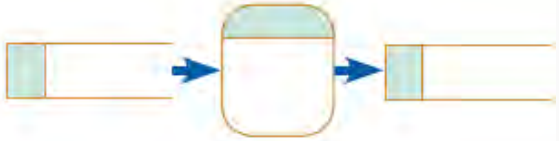
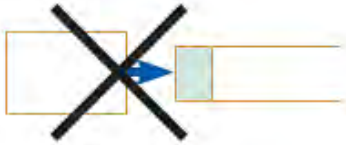
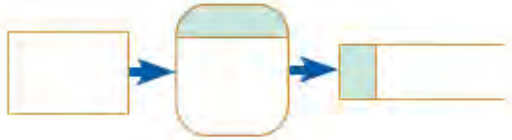
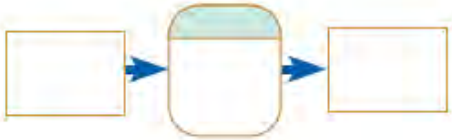
F. Data cannot move directly to an outside sink from a data store. Data must be moved by a process.

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.

Rule	Incorrect	Correct
A.		
B.		
D.		
E.		
F.		
H.		

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.

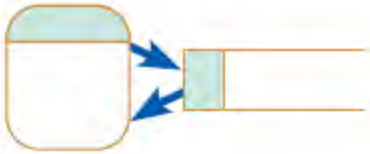
M. A data flow cannot go directly back to the same process it leaves. There must be at least one other process that handles the data flow, produces some other data flow, and returns the original data flow to the beginning process.

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

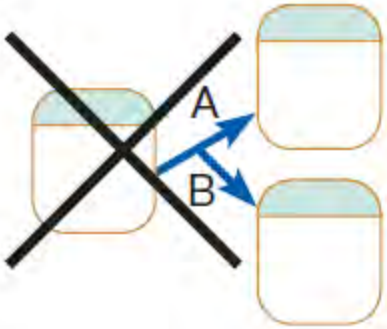
O. A data flow from a data store means retrieve or use.

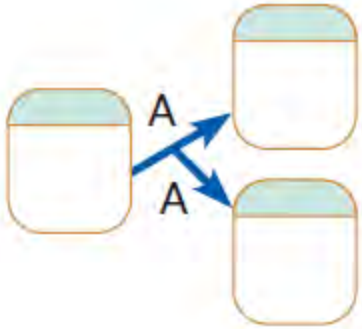
P. A data flow has a noun phrase label. More than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

J.

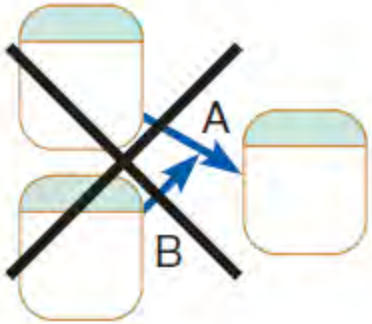


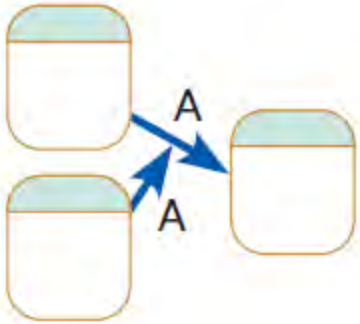
K.



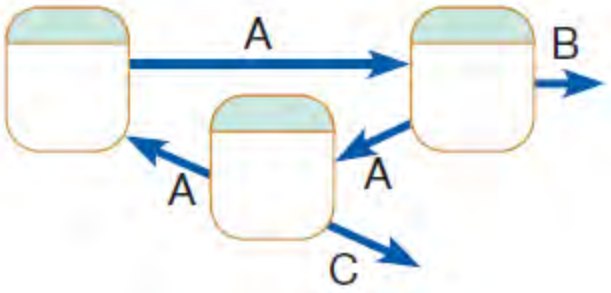


L.





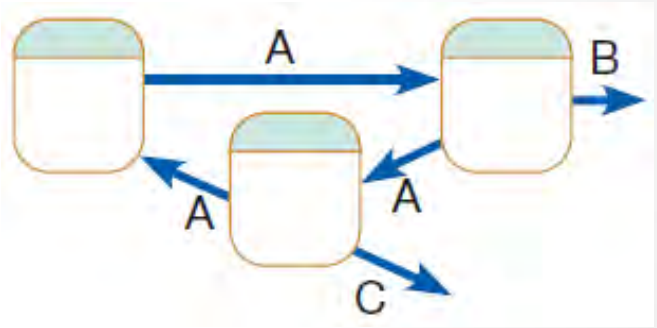
M.



Data Flow Diagramming Rules - cont.

Additional Rules

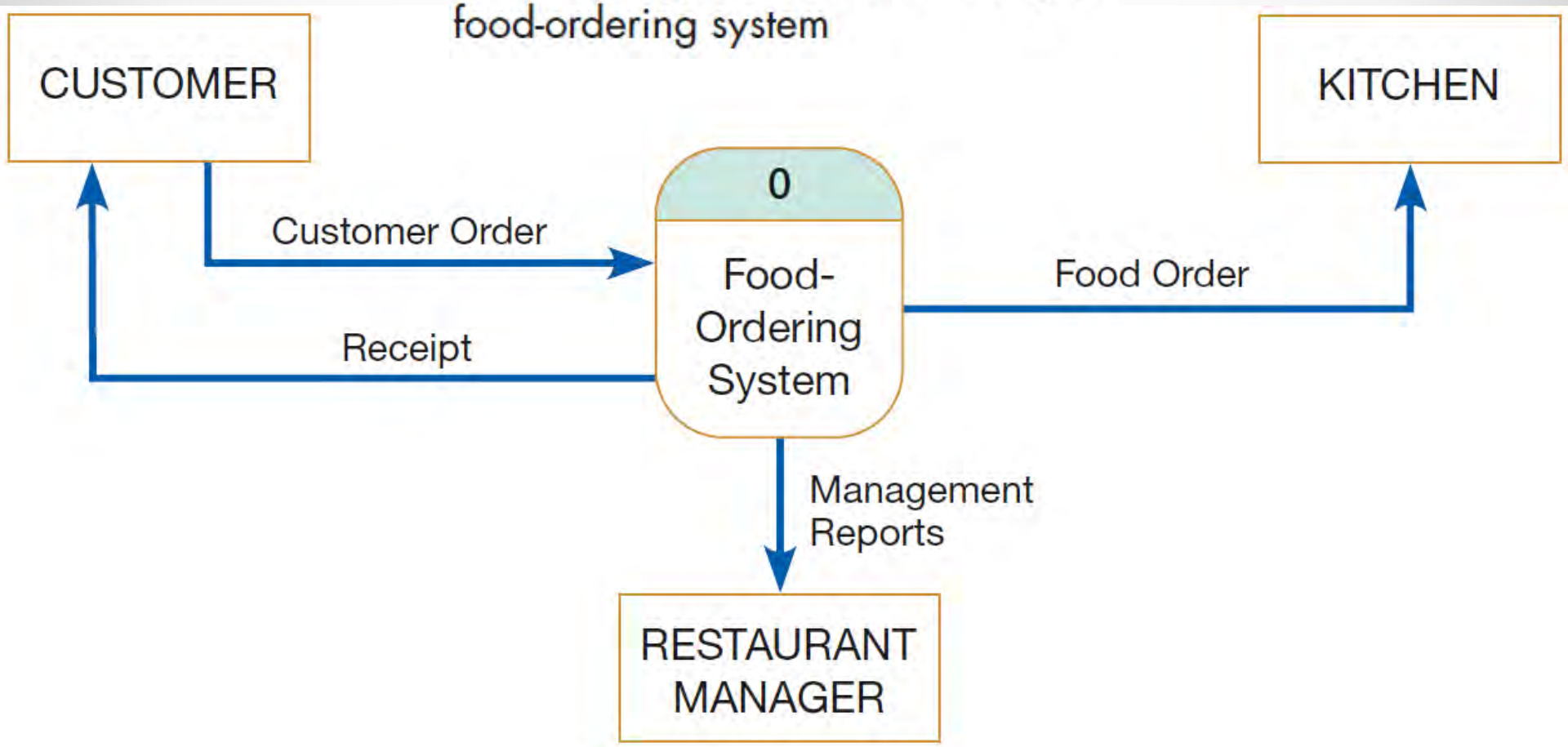
- The inputs to a process are *different* from the outputs of that process
  - exception: same input goes in and out of a process, but the process also produces other new data flows

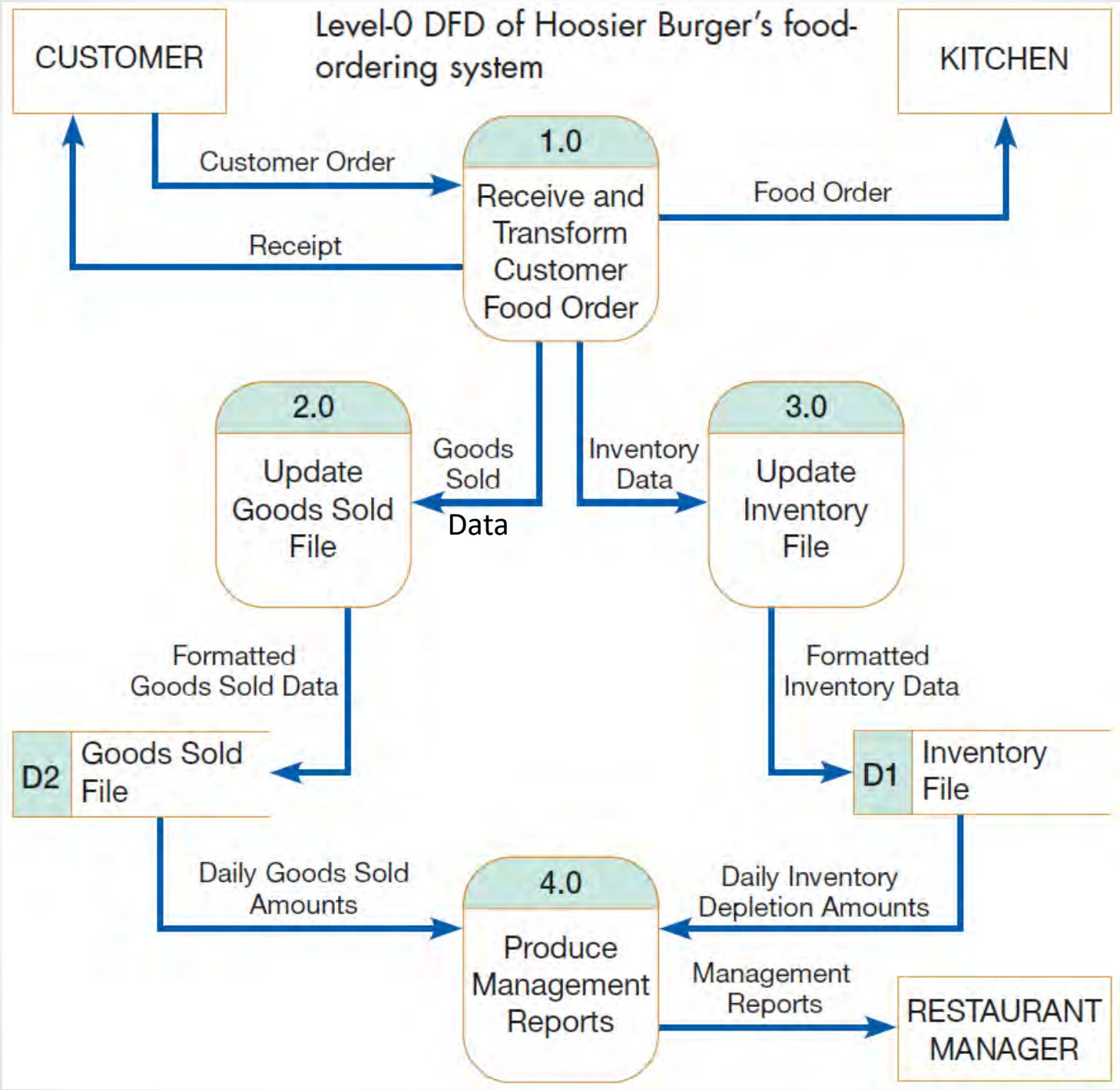


- Objects on a DFD have *unique names*
  - however, you may repeat data stores and sources/sinks
  - also, when two arrows have the same data flow name, you must be careful that these flows are exactly the same

Decomposition of DFDs

Context diagram of Hoosier Burger's food-ordering system




Level-0 DFD of Hoosier Burger's food-ordering system
CUSTOMER
KITCHEN
1.0
Receive and Transform Customer Food Order
Customer Order
Receipt
Food Order
2.0
Update Goods Sold File
3.0
Update Inventory File
Goods Sold Data
Inventory Data
Formatted Goods Sold Data
Formatted Inventory Data
D2
Goods Sold File
D1
Inventory File
Daily Goods Sold Amounts
Daily Inventory Depletion Amounts
4.0
Produce Management Reports
Management Reports
RESTAURANT MANAGER

Developing DFDs: Level 1

Level-1 diagram:

- 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
- Each level-1, -2, or -*n* DFD represents one process on a level-'*n*-1' DFD
- Each DFD should be on a separate page
- No DFD should have more than ~7 processes (too many processes makes the diagram too crowded/ difficult to understand)

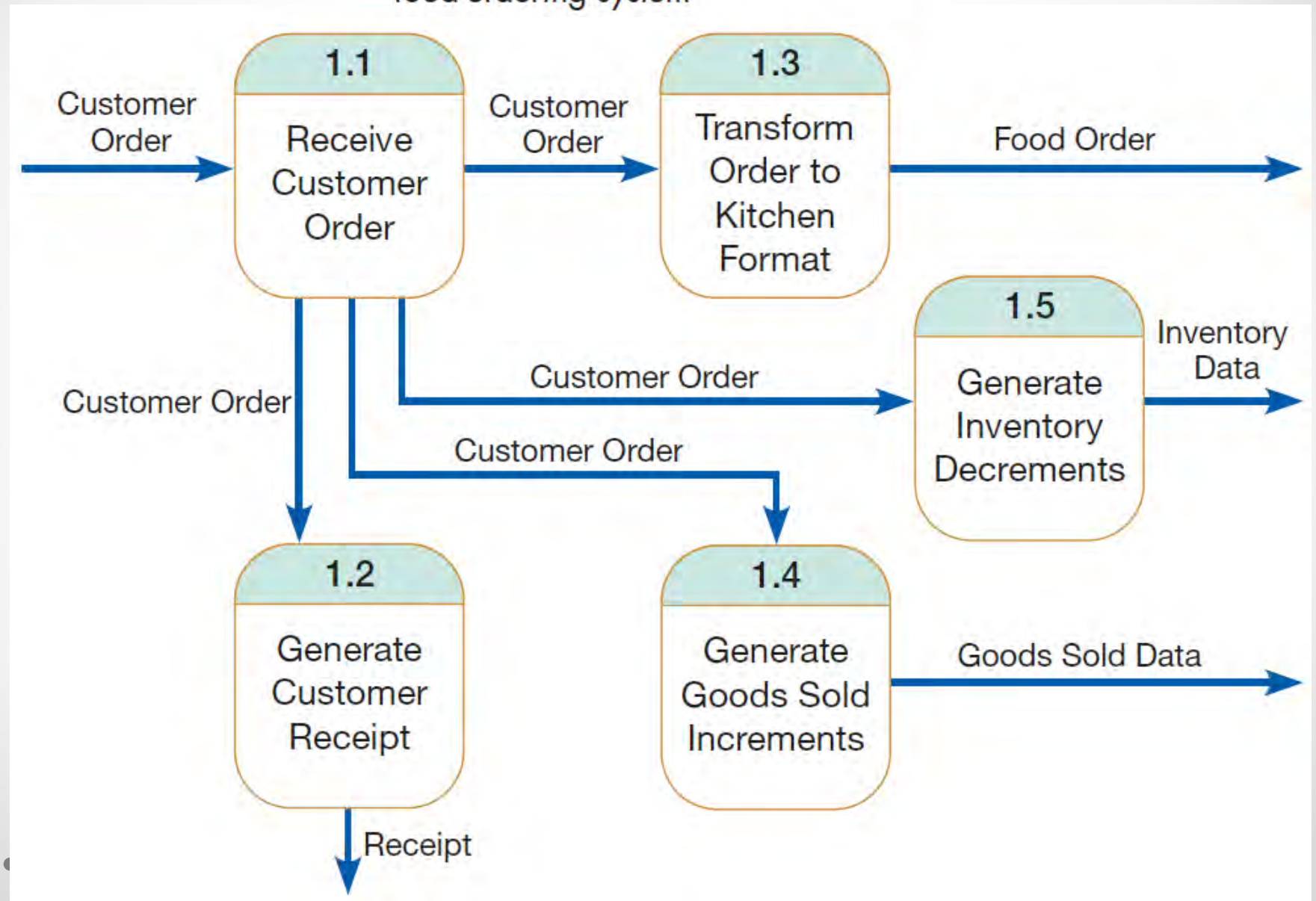
Developing DFDs: Level 1

Level-1 diagram (cont.):

- Process 1.0 decomposition:
  - Each of the five processes is labeled as a *subprocess* of Process 1.0: Process 1.1, Process 1.2, etc.
  - Notice how *no sources or sinks* are represented (only for context and level-0 DFDs)

- Note regarding Processes 2.0 and 3.0:
  - Both perform similar functions since both use data input to *update data stores*
  - However, updating a data store is a *singular logical function* $\Rightarrow$ no need to decompose them further

Level-1 diagram showing the decomposition of Process 1.0 from the level-0 diagram for Hoosier Burger's food-ordering system



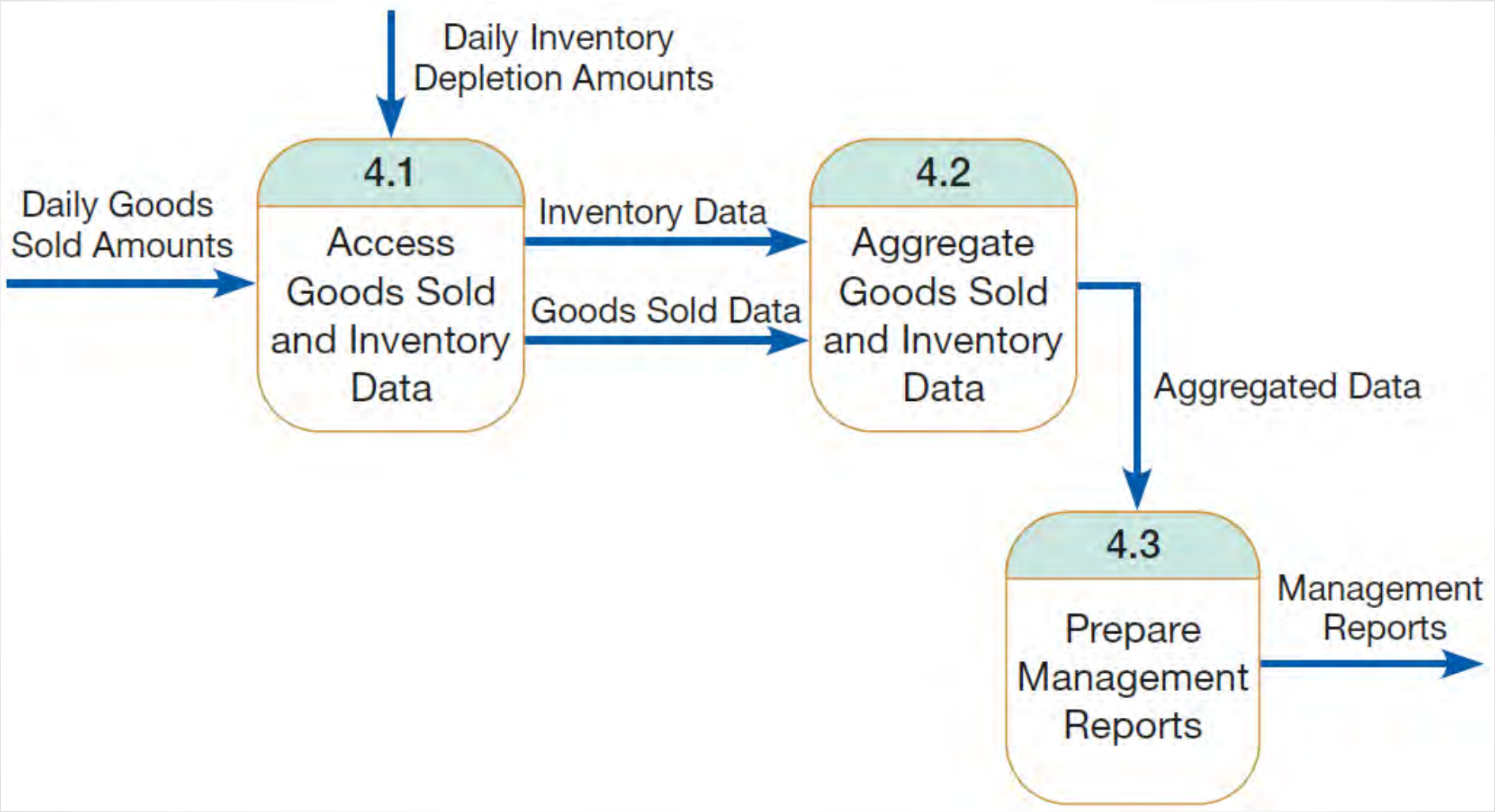
Developing DFDs: Level 1

Level-1 diagram (cont.):

Process 4.0 decomposition:

- We can decompose Process 4.0 into at least three subprocesses:
  - Process 4.1: Access Goods Sold and Inventory Data
  - Process 4.2: Aggregate Goods Sold and Inventory Data
  - Process 4.3: Prepare Management Reports

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



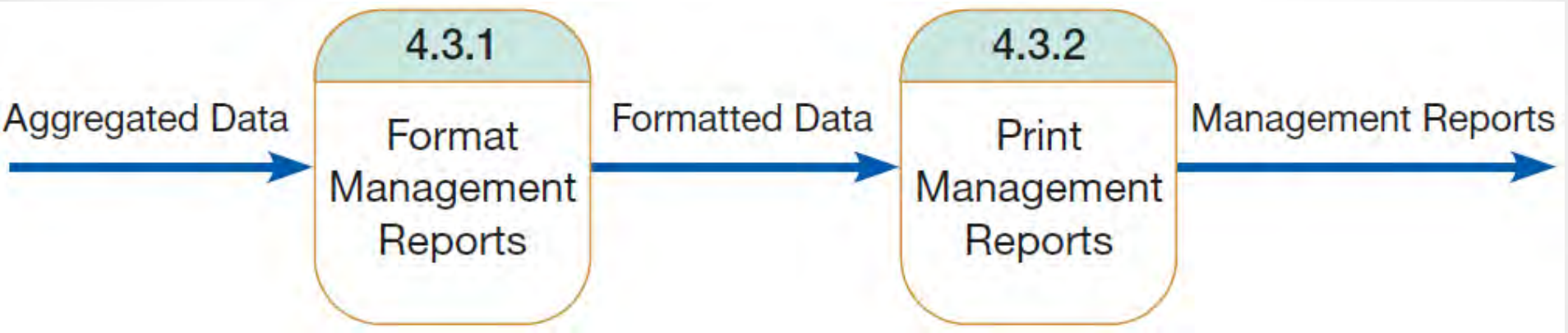
Developing DFDs: Level 2

Level-2 diagram:

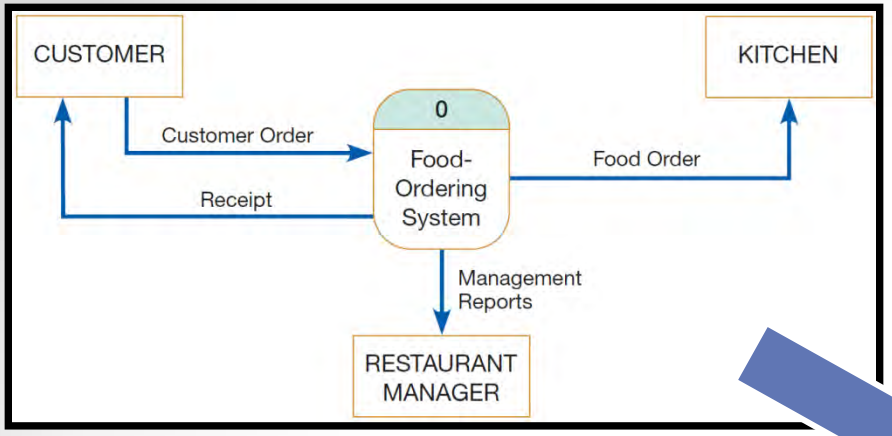
Process 4.3 decomposition:

- We can decompose Process 4.3 into 2 subprocesses:
  - Process 4.3.1: Format Management Reports
  - Process 4.3.2: Print Management Reports
- Note about action verbs used in DFD processes
  - e.g.: *Receive*, *Calculate*, *Transform*, *Generate*, *Produce*
  - these process names often are the same as the verbs used in many computer programming languages, e.g. *Merge*, *Sort*, *Read*, *Write*, *Print*
  - $\Rightarrow$ process names should be short and descriptive of process

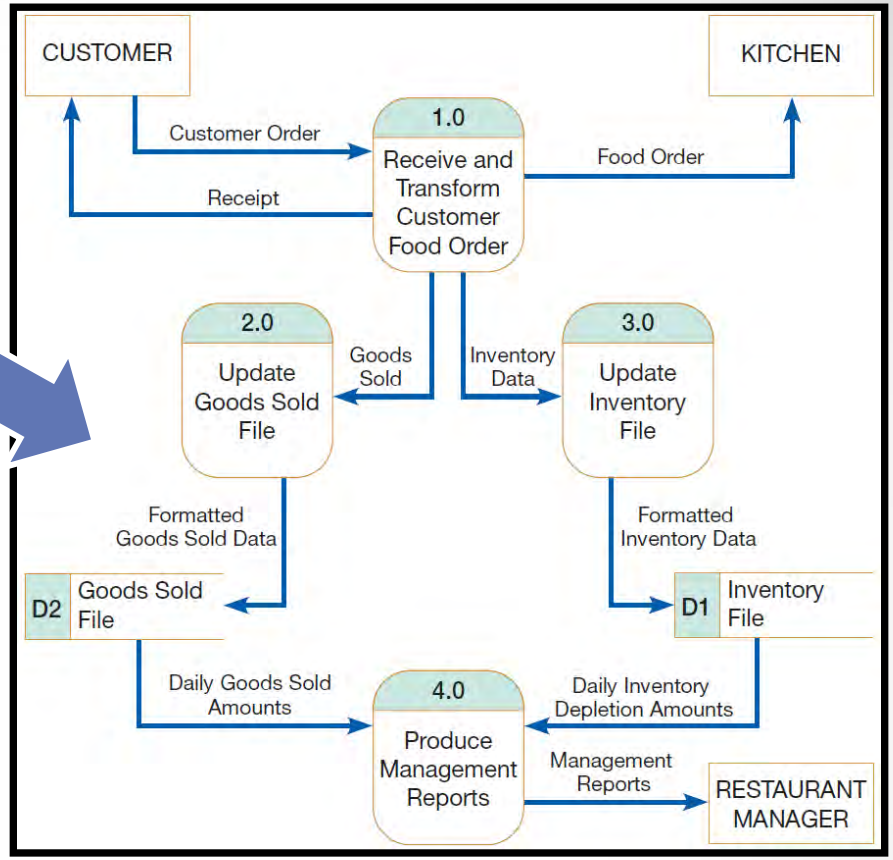
Level-2 diagram showing the decomposition of Process 4.3 from the level-1 diagram for Process 4.0 for Hoosier Burger's food-ordering system



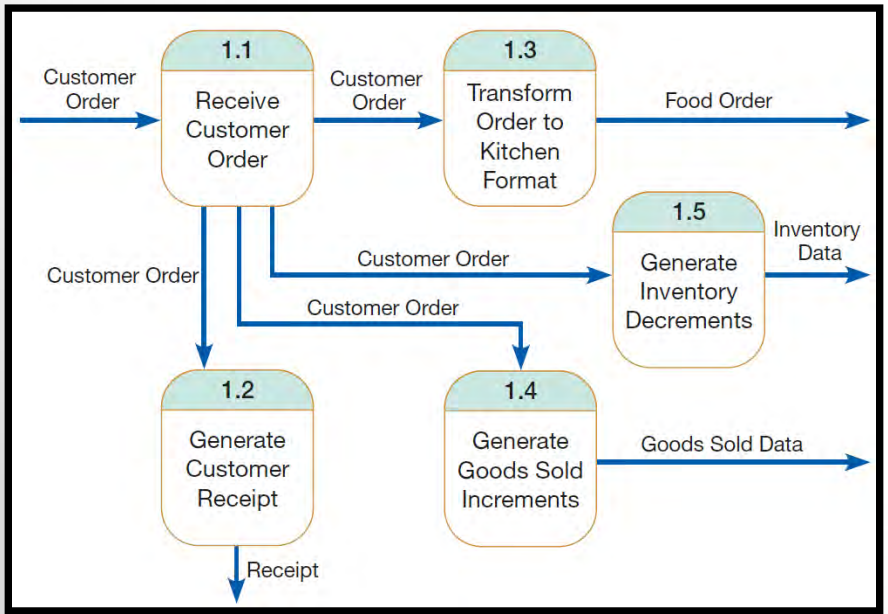
Food Ordering System Summary



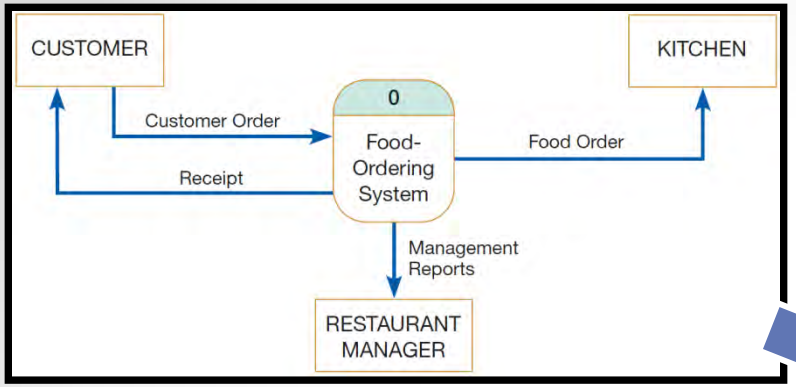
Context diagram of Hoosier Burger's food-ordering system



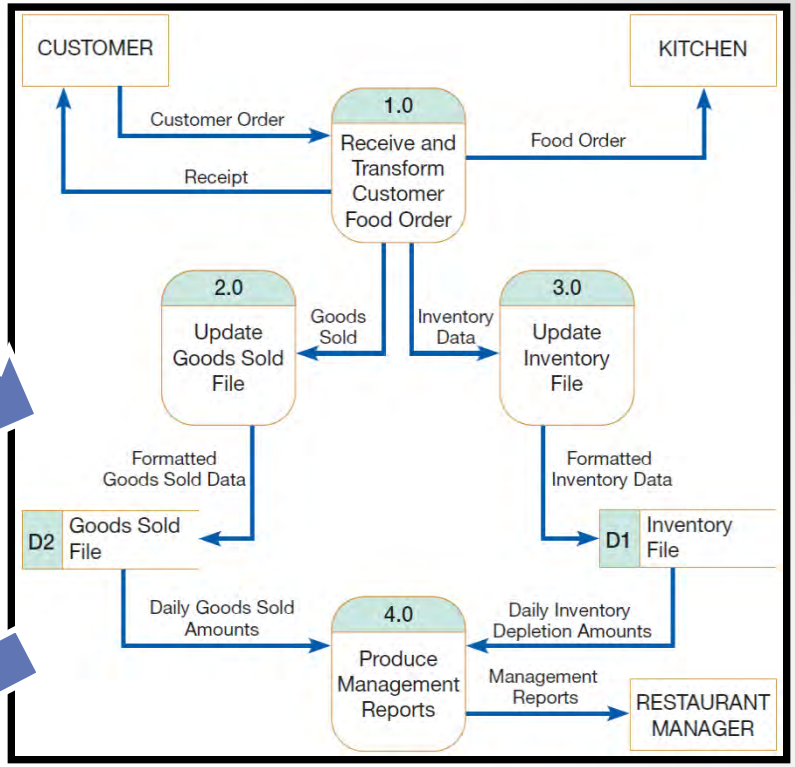
Level-0 DFD of Hoosier Burger's food-ordering system



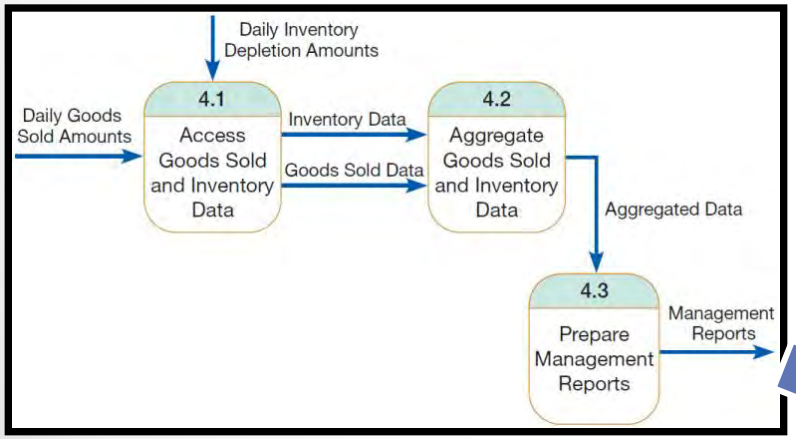
Level-1 diagram showing the decomposition of Process 1.0 from the level-0 diagram for Hoosier Burger's food-ordering system



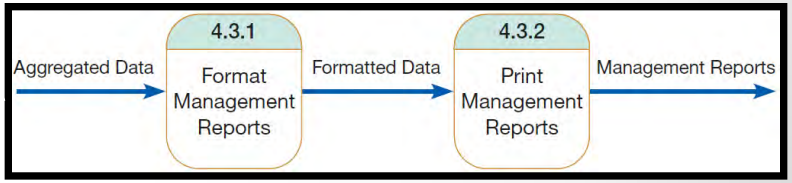
Context diagram of Hoosier Burger's food-ordering system



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



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



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

Balancing DFDs

Balancing DFDs

Balancing DFDs

- Balancing: conservation of I's / O's to a DFD process when that process is decomposed to a lower level

- Process in context diagram must have same I's / O's when decomposed into a level-0 diagram, e.g.:

  Context Diagram:

  - 1 I: customer order (from customer)
  - 3 Os: customer receipt, food order (for kitchen), management reports

  Level-0 Diagram:

  - note *the same* 1 I + 3 O's
  - also, no new I's to or O's from the system have been added

  ⇒ context diagram and level-0 DFDs are *balanced*

Balancing DFDs (cont.)

Balancing DFDs (cont.)

- Process in level-0 diagram must have same I's / O's when decomposed into a level-1 diagram, e.g.:

  Process 1.0 in Level-0 Diagram:

  o 1 I + 4 O's

  Process 1.0 decomposition in Level-1 Diagram:

  o note the same single I and multiple O's

  o also no new I's or O's have been introduced

  ⇒ again we see conservation/balance of I's and O's

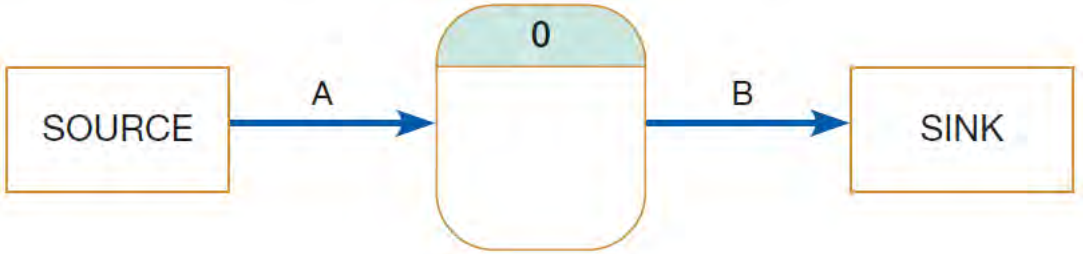
- Can you also compare Process 4.0 with its decomposition? Are they balanced?

Balancing DFDs (cont.)

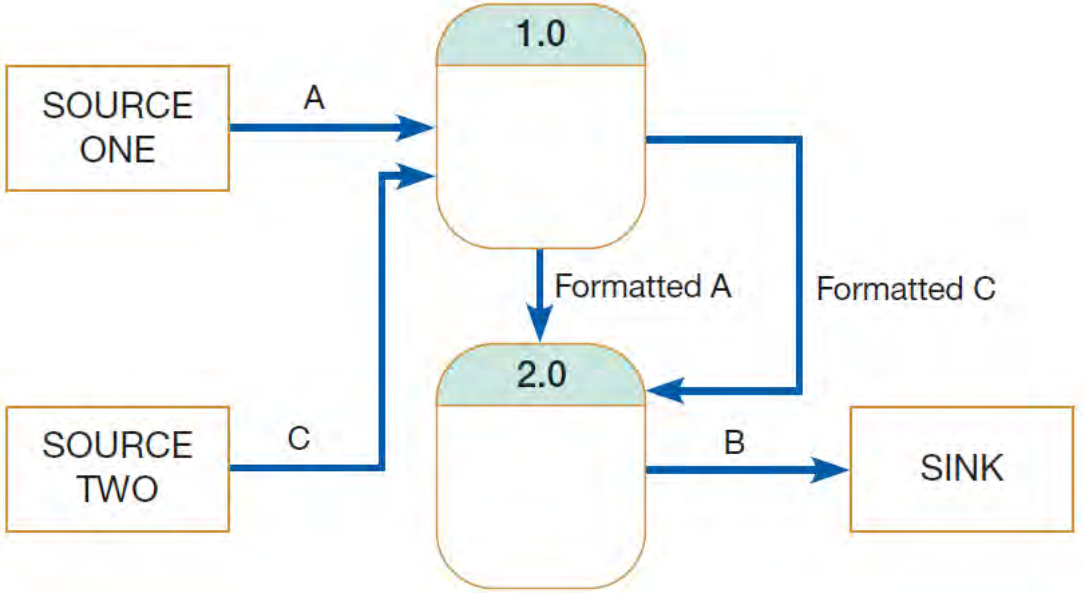
Balancing DFDs (cont.)

- An *unbalanced* set of DFDs (why? can we fix this?)

(a) Context diagram



(b) Level-0 diagram



Balancing DFDs (cont.)

Data flow splitting

- Composite (aka package) data flow for a process can be split apart into several subflows

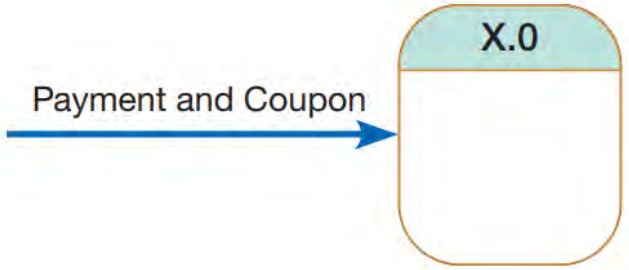
- This occurs when:
  - process is decomposed (aka exploded/nested) into 2/more subprocesses (see e.g.)
  - each subprocess (e.g. Level-1) receives one of the components of the composite data flow from the higher-level DFD (e.g. Level-0)

- Note, these diagrams are still balanced because exactly the *same data* are included in each diagram

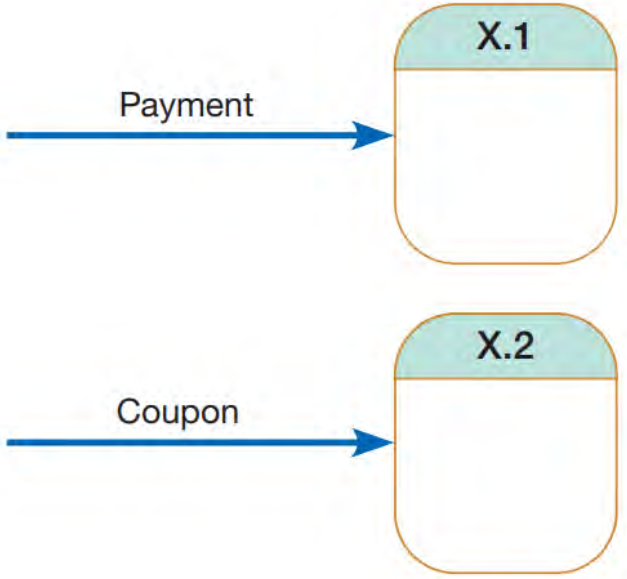
Balancing DFDs (cont.)

Data flow splitting (cont.)

FIGURE 7-11
Example of data flow splitting
(a) Composite data flow



(b) Disaggregated data flows



Balancing DFDs (cont.)

Data flow splitting (cont.)

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:* Based on Celko, 1987.)

Guidelines for Drawing DFDs

Guidelines for Drawing DFDs

Guidelines for Drawing DFDs:

1. Completeness
2. Consistency
3. Timing considerations
4. Iterative nature of drawing DFDs
5. Primitive DFDs

Note, these are considered besides DFD diagramming rules (A – T)

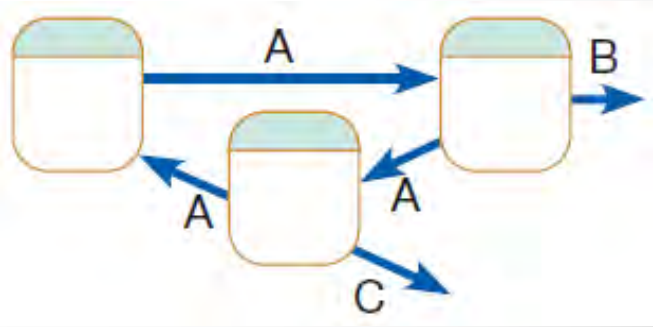
Guidelines for Drawing DFDs (cont.)

1. Completeness:

- It is extent to which all necessary components of a DFD have been included and fully described

- Examples of incomplete DFDs
  - o data flows that do not lead anywhere
  - o data stores, processes, or external entities that are not connected to anything else

- CASE tools have built-in facilities that you can run to help you determine if your DFD is incomplete and/or contains errors/problems



Guidelines for Drawing DFDs (cont.)

2. Consistency:

- It is extent to which information contained on one level of a set of nested DFDs is also included on other levels

- Examples of inconsistency:
  - level-1 diagram with no level-0 diagram
  - data flow that appears on a higher-level DFD but not on lower levels (this's also violation of balancing)
  - data flow attached to one object on a lower-level diagram but also attached to another object at a higher level e.g. data flow named Payment, which serves as input to *Process 1* on a level-0 DFD, appears as input to *Process 2.1* on a level-1 diagram for Process 2

- Note CASE tools can also be used to remove such inconsistencies*

Guidelines for Drawing DFDs (cont.)

3. Timing considerations:

- DFDs do not do a very good job of representing time
- There is *no indication* of whether a data flow occurs constantly in real time, once per week, etc.
- Also, there is no indication of when a system would run
- ⇒ you should draw DFDs as if the system you are modeling has *never started* and will *never stop* (i.e. *timeless*)

Guidelines for Drawing DFDs (cont.)

4. Iterative nature of drawing DFDs:

- First DFD you draw will rarely capture perfectly the system you are modeling
- $\Rightarrow$ you should be willing to revise/repeat DFDs several times (i.e. in an iterative fashion)
- With each attempt, you will come closer to a good approximation of the system you are modeling
- It should take you ~3 revisions for each DFD you draw
- Note, CASE tools make revising drawings much easier (than manual drawing)

Guidelines for Drawing DFDs (cont.)

5. Primitive DFDs:

- A difficult decision when drawing DFDs is: when to stop decomposing processes? i.e. how to determine if you have reached the *lowest logical level*?

- Complete sets of DFDs should extend to the primitive level (i.e. lowest level of decomposition for a DFD)
  - where every component reflects certain *irreducible* properties
  - e.g. process represents a single database operation (e.g. retrieve, update, create, delete, or read) and
  - every data store (e.g. customer, employee, product, order) represents data about a single entity

Videos to Watch

- What is DFD? Data Flow Diagram Symbols and More https://youtu.be/6VGTvgaJllM (*Smartdraw*)

- How to Draw Data Flow Diagram? https://youtu.be/ztZsEI6C-mI (*Visual Paradigm*)

- DFD Diagram 0 https://youtu.be/Ik85hZkyYPA (*Visible Analyst*)

Sources

- Modern Systems Analysis and Design. Joseph S. Valacich and Joey F. George. Pearson. Eighth Ed. 2017. Chapter 7.
- Design of Industrial Information Systems. Thomas Boucher, and Ali Yalcin. Academic Press. First Ed. 2006. Chapter 4.