Modern Systems Analysis and Design

Fourth Edition

Jeffrey A. Hoffer
Joey F. George
Joseph S. Valacich

Structuring System Process Requirements

Learning Objectives

- ✓ Understand logical process modeling via data flow diagrams (DFDs).
- ✓ Draw DFDs of well structured process models.
- ✓ Decompose DFDs into lower-level diagrams.
- ✓ Balance high-level and low-level DFDs.
- ✓ Explain differences between current physical, current logical, new physical, and new logical DFDs.
- ✓ Use DFDs for analyzing information systems.
- ✓ Explain use cases and use case diagrams.

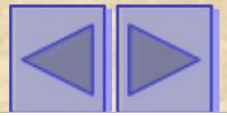
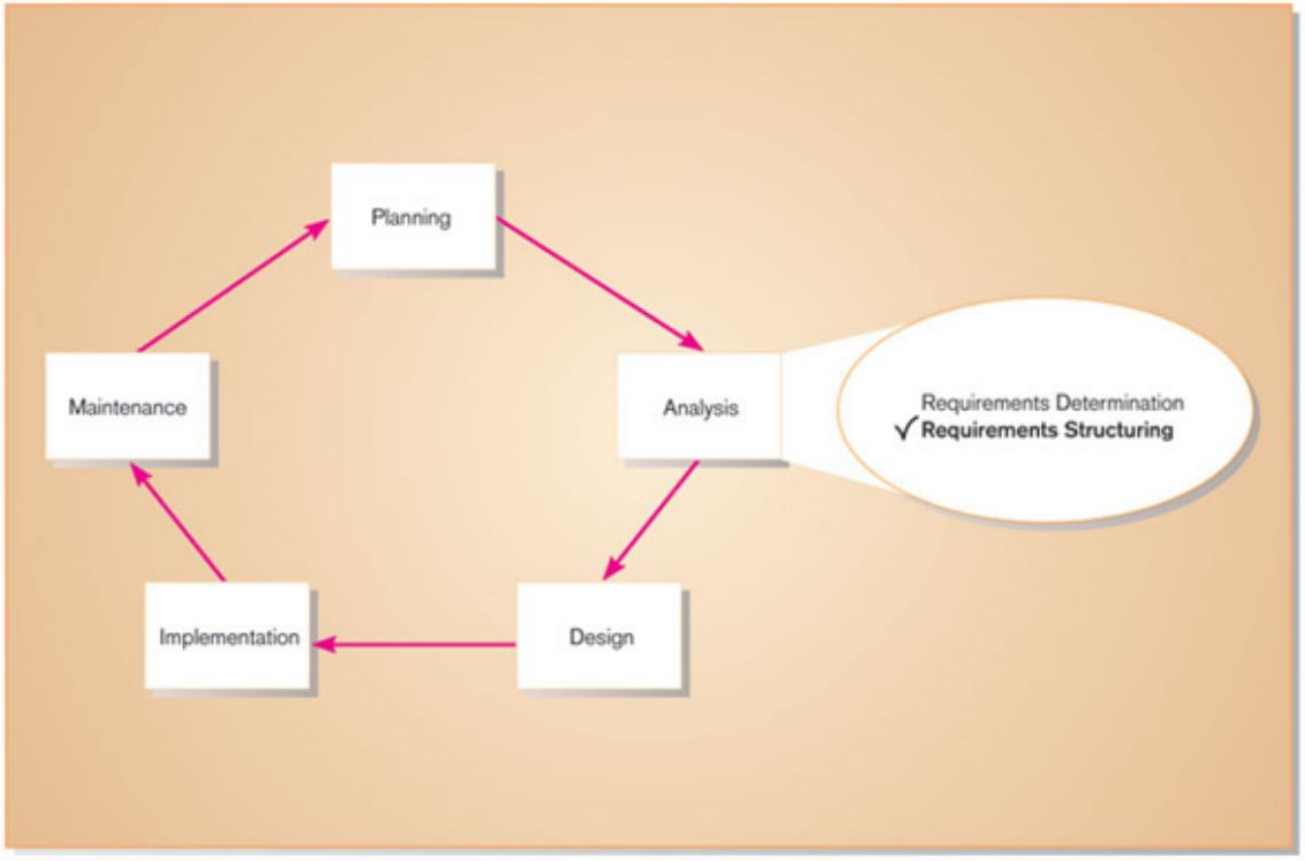


Figure 7-1 Systems development life cycle with the analysis phase highlighted

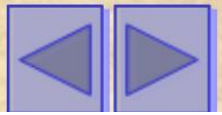


Process Modeling

- Graphically represent the processes that capture, manipulate, store, and distribute data between a system and its environment and among system components
- Utilize information gathered during requirements determination
- Processes and data structures are modeled

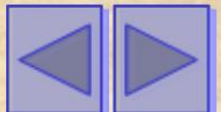
Process Modeling (cont.) Deliverables and Outcomes

- Context data flow diagram (DFD)
  - Scope of system
- DFDs of current physical and logical system
  - Enables analysts to understand current system
- DFDs of new logical system
  - Technology independent
  - Show data flows, structure, and functional requirements of new system
- Thorough description of each DFD component



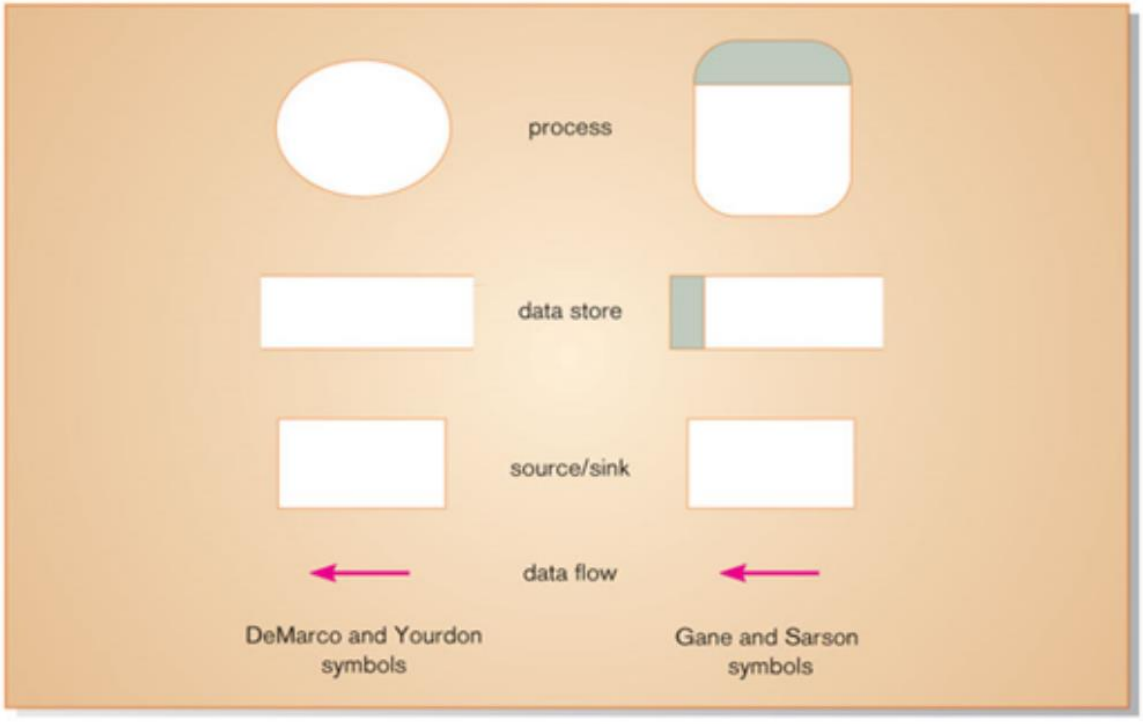
Data Flow Diagram (DFD)

- A picture of the movement of data between external entities and the processes and data stores within a system
- Difference from system flowcharts:
  - DFDs depict logical data flow independent of technology
  - Flowcharts depict details of physical systems



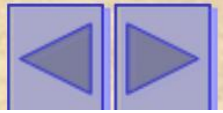
DFD Symbols

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

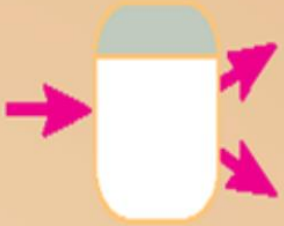


DFD Symbols (cont.)

- Process: work or actions performed on data (inside the system)
- Data store: data at rest (inside the system)
- Source/sink: external entity that is origin or destination of data (outside the system)
- Data flow: arrows depicting movement of data



DFD Diagramming Rules
Process

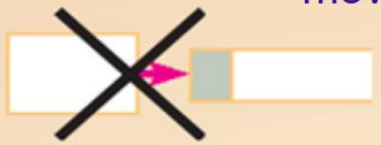


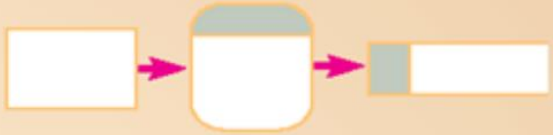
No process can have only outputs or only inputs…processes must have both outputs and inputs.

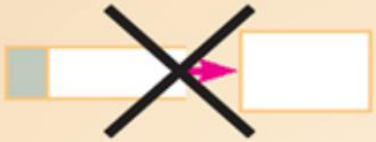
Process labels should be verb phrases.

DFD Diagramming Rules
Data Store

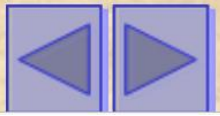
All flows to or from a data store must move through a process.



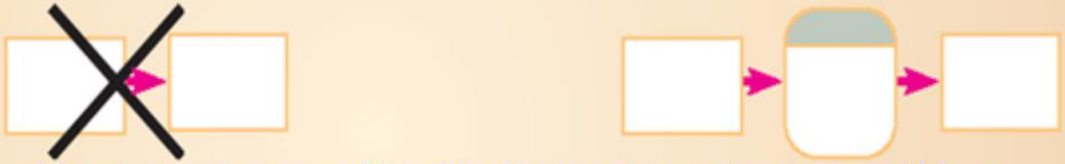




Data store labels should be noun phrases.



DFD Diagramming Rules Source/Sink



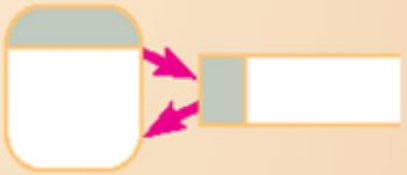
No data moves directly between external entities without going through a process.

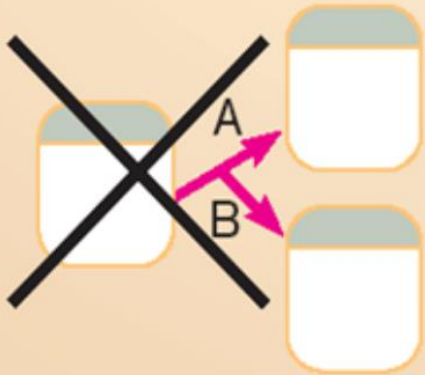
Interactions between external entities without intervening processes are outside the system and therefore not represented in the DFD.

Source and sink labels should be noun phrases.

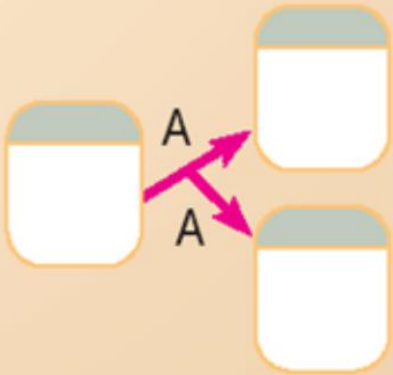
DFD Diagramming Rules
Data Flow

Bidirectional flow between process and data store is represented by two separate arrows.

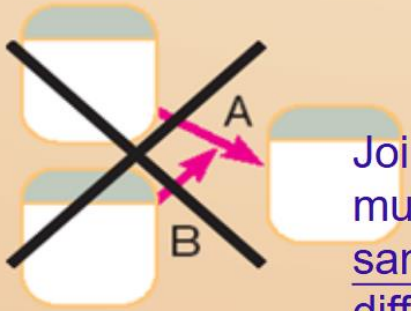




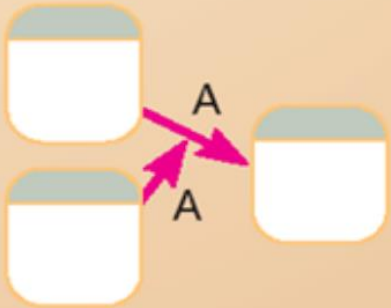
Forked data flow must refer to exact same data item (not different data items) from a common location to multiple destinations.



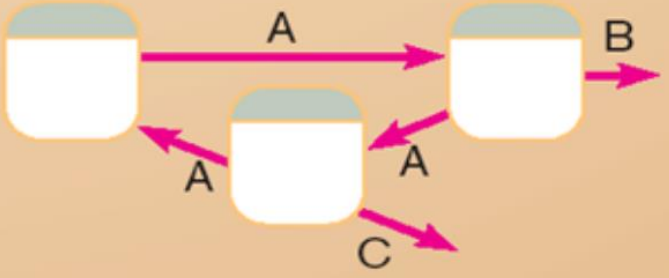
DFD Diagramming Rules



Joined data flow must refer to exact same data item (not different data items) from multiple sources to a common location.

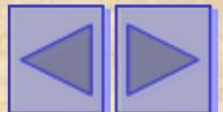


Data flow cannot go directly from a process to itself, must go through intervening processes.



DFD Diagramming Rules
Data Flow (cont.)

- Data flow from a process to a data store means update (insert, delete or change).
- Data flow from a data store to a process means retrieve or use.
- Data flow labels should be noun phrases.

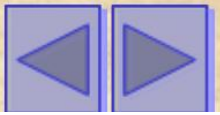


Functional Decomposition

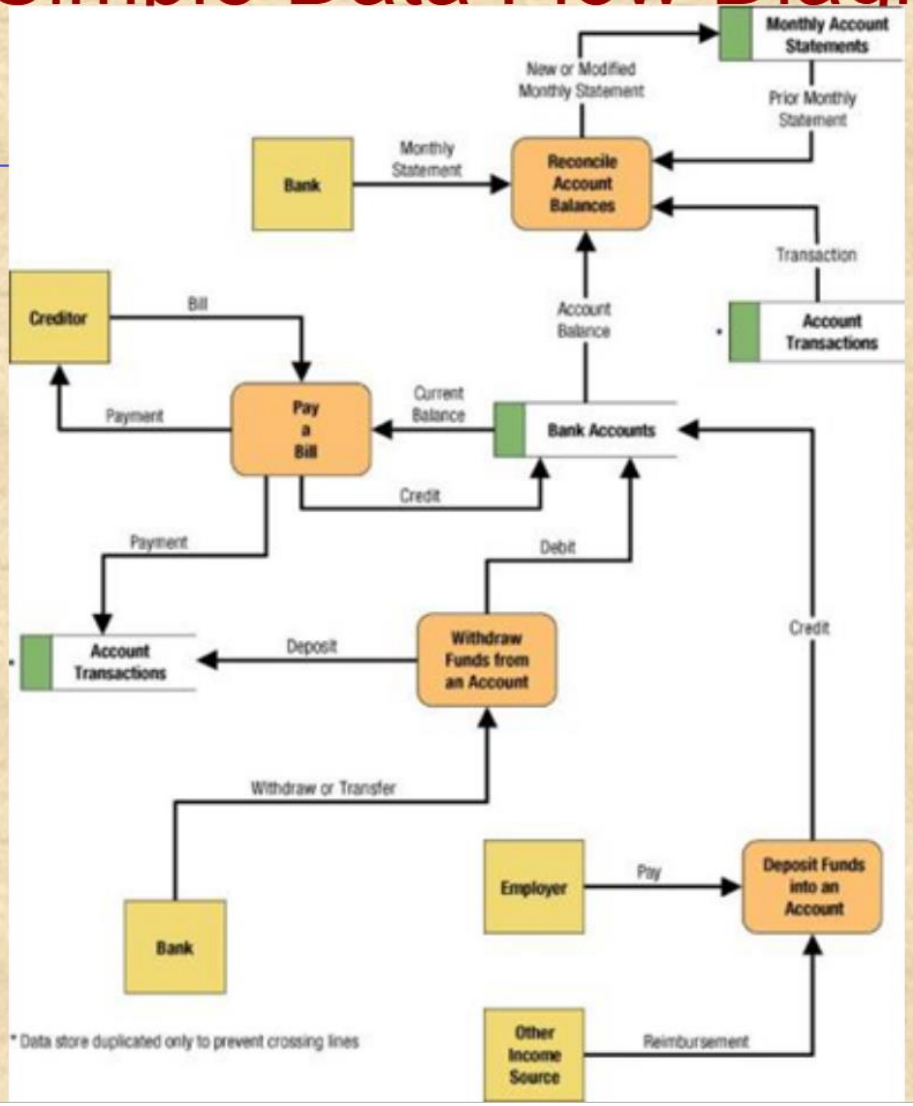
- An iterative process of breaking a system description down into finer and finer detail
- High-level processes described in terms of lower-level sub-processes
- DFD charts created for each level of detail

DFD Levels

- Context DFD
  - Overview of the organizational system
- Level-0 DFD
  - Representation of system's major processes at high level of abstraction
- Level-1 DFD
  - Results from decomposition of Level 0 diagram
- Level-*n* DFD
  - Results from decomposition of Level *n*-1 diagram

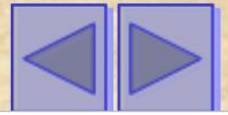


Simple Data Flow Diagram



Differences Between DFDs and Flowcharts

- Processes on DFDs can operate in parallel (at-the-same-time)
  - Processes on flowcharts execute one at a time
- DFDs show the flow of data through a system
  - Flowcharts show the flow of control (sequence and transfer of control)
- Processes on a DFD can have dramatically different timing (daily, weekly, on demand)
  - Processes on flowcharts are part of a single program with consistent timing

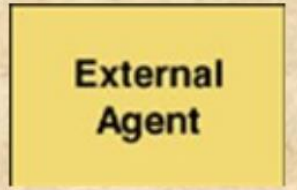


External Agents

External agent – an outside person, unit, system, or organization that interacts with a system. Also called an *external entity*.

- External agents define the "boundary" or scope of a system being modeled.
- As scope changes, external agents can become processes, and vice versa.
- Almost always one of the following:
  - Office, department, division.
  - An external organization or agency.
  - Another business or another information system.
  - One of system's end-users or managers
- Named with descriptive, singular noun

Gane and Sarson shape

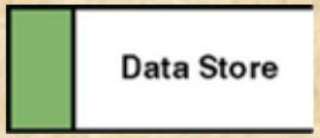


DeMarco/Yourdon shape

Data Stores

Data store – stored data intended for later use. Synonyms are *file* and *database*.

- Frequently implemented as a file or database.
- A data store is "data at rest" compared to a data flow that is "data in motion."
- Almost always one of the following:
  - Persons (or groups of persons)
  - Places
  - Objects
  - Events (about which data is captured)
  - Concepts (about which data is important)
- Data stores depicted on a DFD store all instances of data entities (depicted on an ERD)
- Named with plural noun



Gane and Sarson shape

DeMarco/Yourdon shape

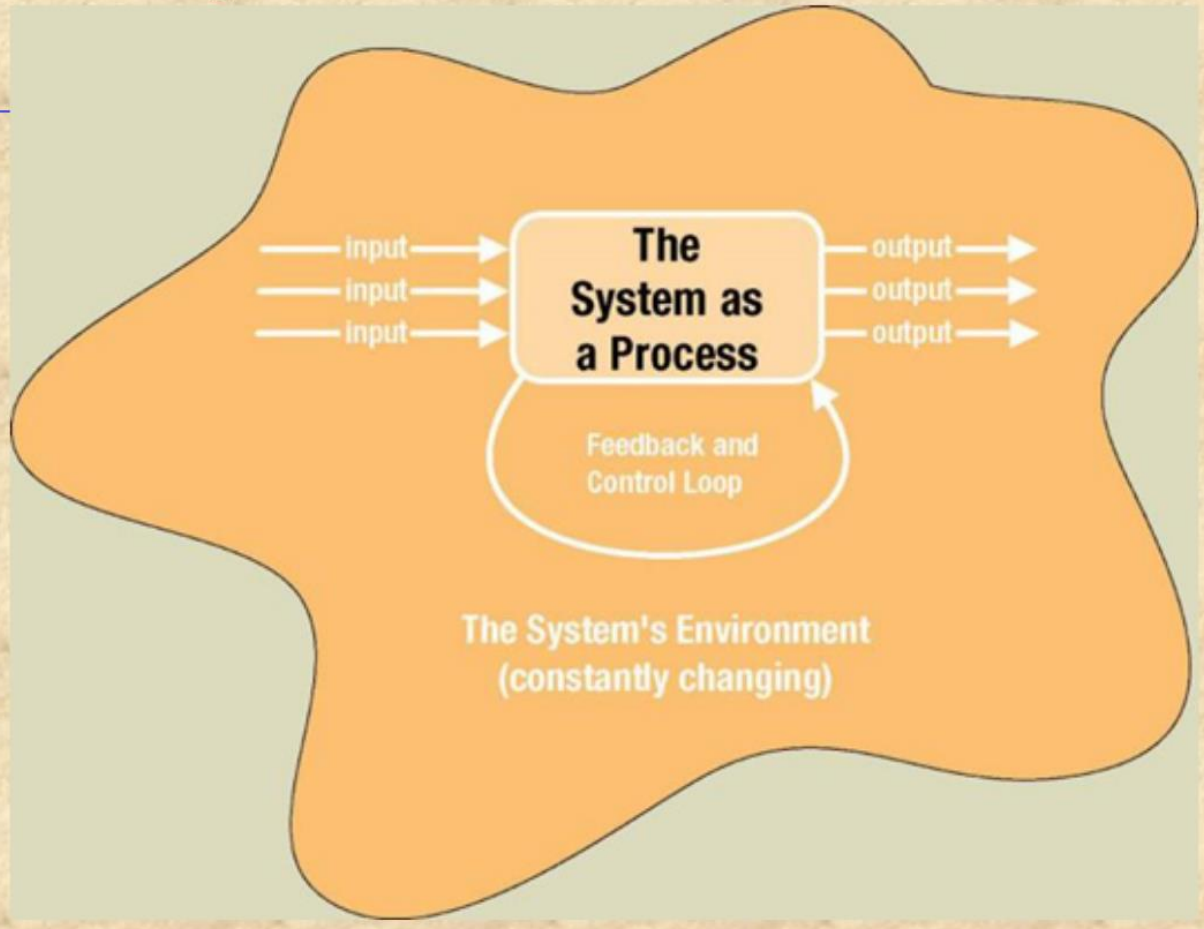
Process Concepts

Process – work performed by a system in response to incoming data flows or conditions. A synonym is *transform*.

- All information systems include processes - usually many of them
- Processes respond to business events and conditions and transform data into useful information
- Modeling processes helps us to understand the interactions with the system's environment, other systems, and other processes.
- Named with a strong action verb followed by object clause describing what the work is performed on/for .

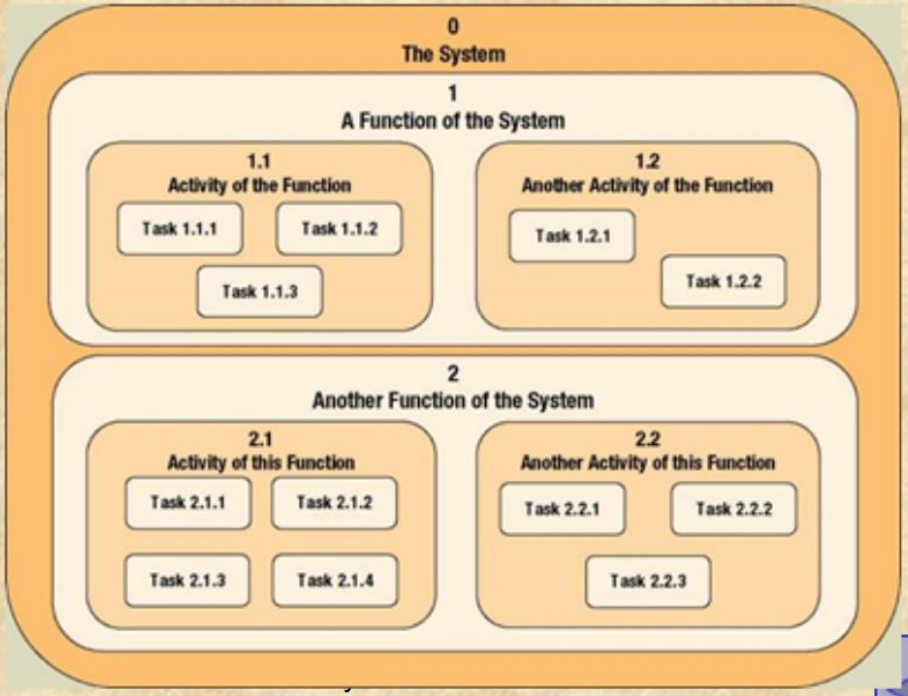
Gane and Sarson shape

The System is Itself a Process



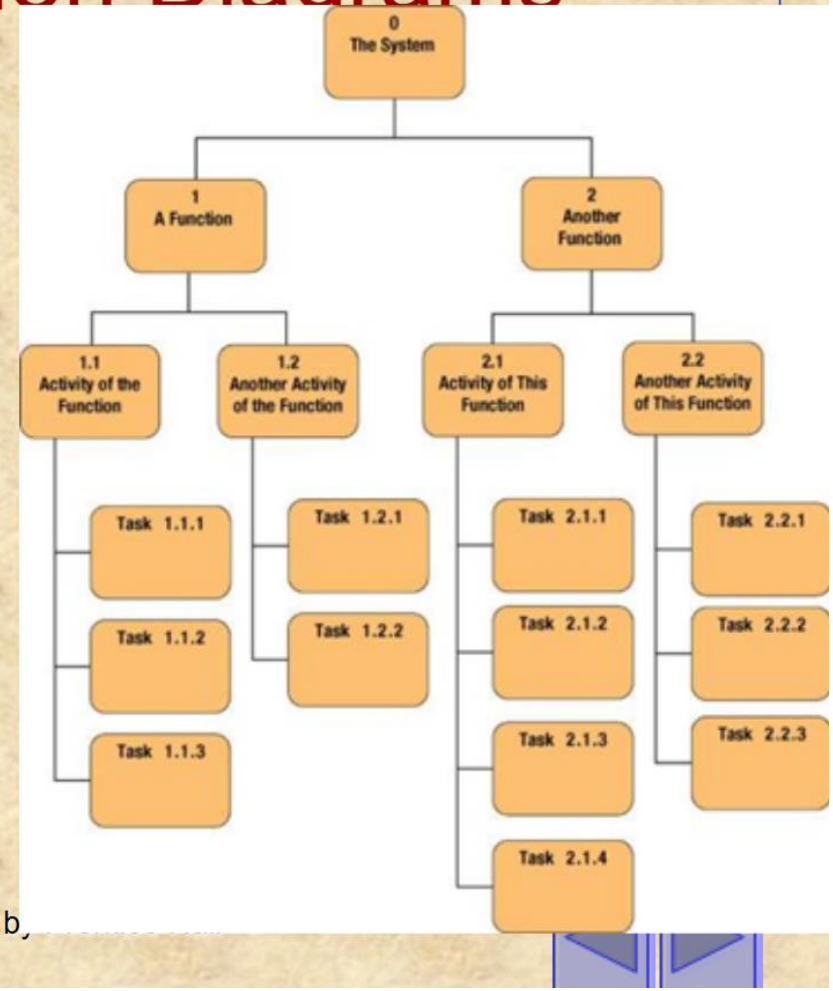
Process Decomposition

Decomposition – the act of breaking a system into sub-components. Each level of abstraction reveals more or less detail.



Decomposition Diagrams

Decomposition diagram – a tool used to depict the decomposition of a system. Also called hierarchy chart.



Types of Logical Processes

Function – a set of related and ongoing activities of a business.

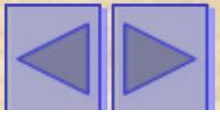
- A function has no start or end.

Event – a logical unit of work that must be completed as a whole. Sometimes called a *transaction*.

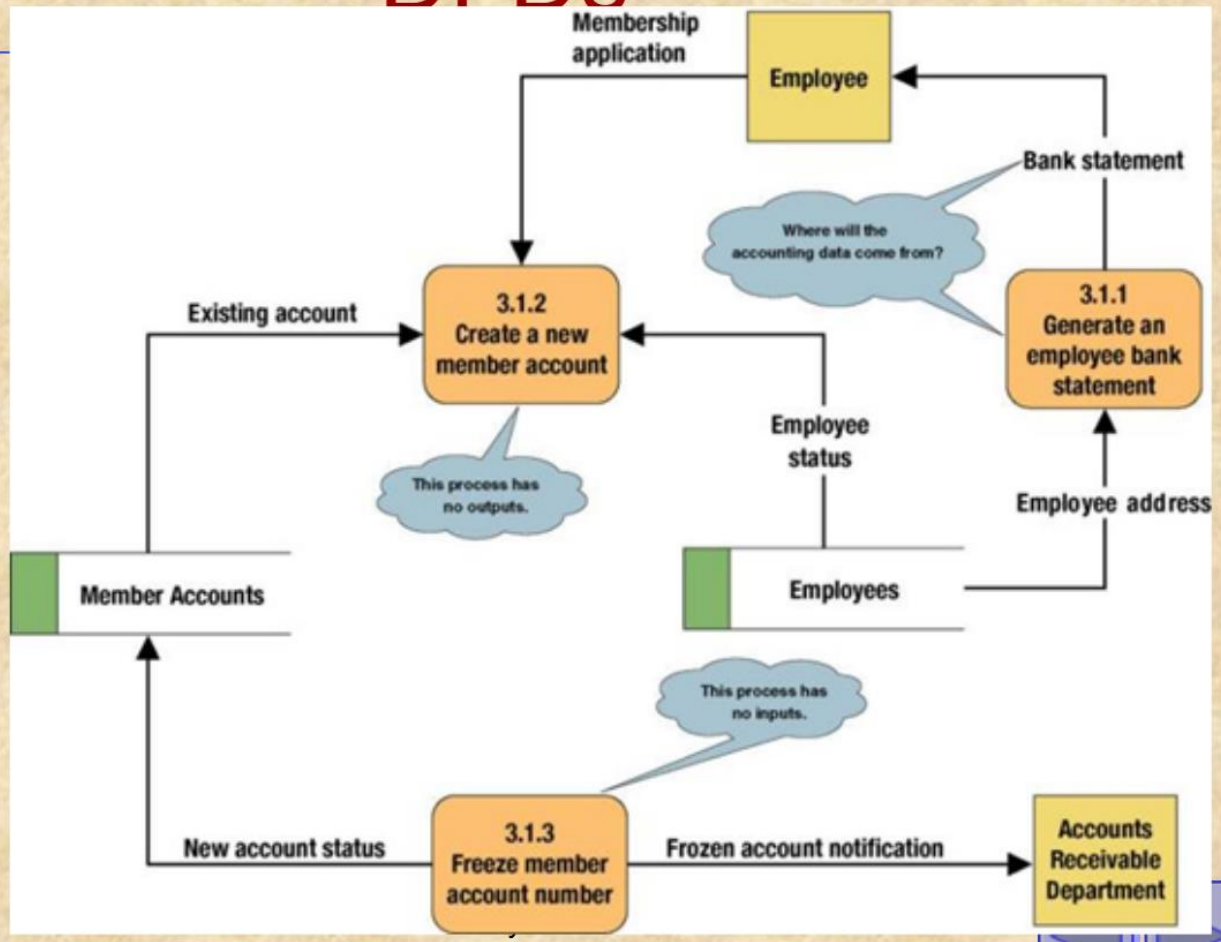
- Triggered by a discrete input and is completed when process has responded with appropriate outputs.
- Functions consist of processes that respond to events.

Elementary process – a discrete, detailed activity or task required to complete the response to an event. Also called a *primitive process*.

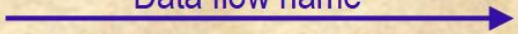
- The lowest level of detail depicted in a process model.

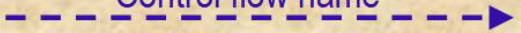


Common Process Errors on DFDs



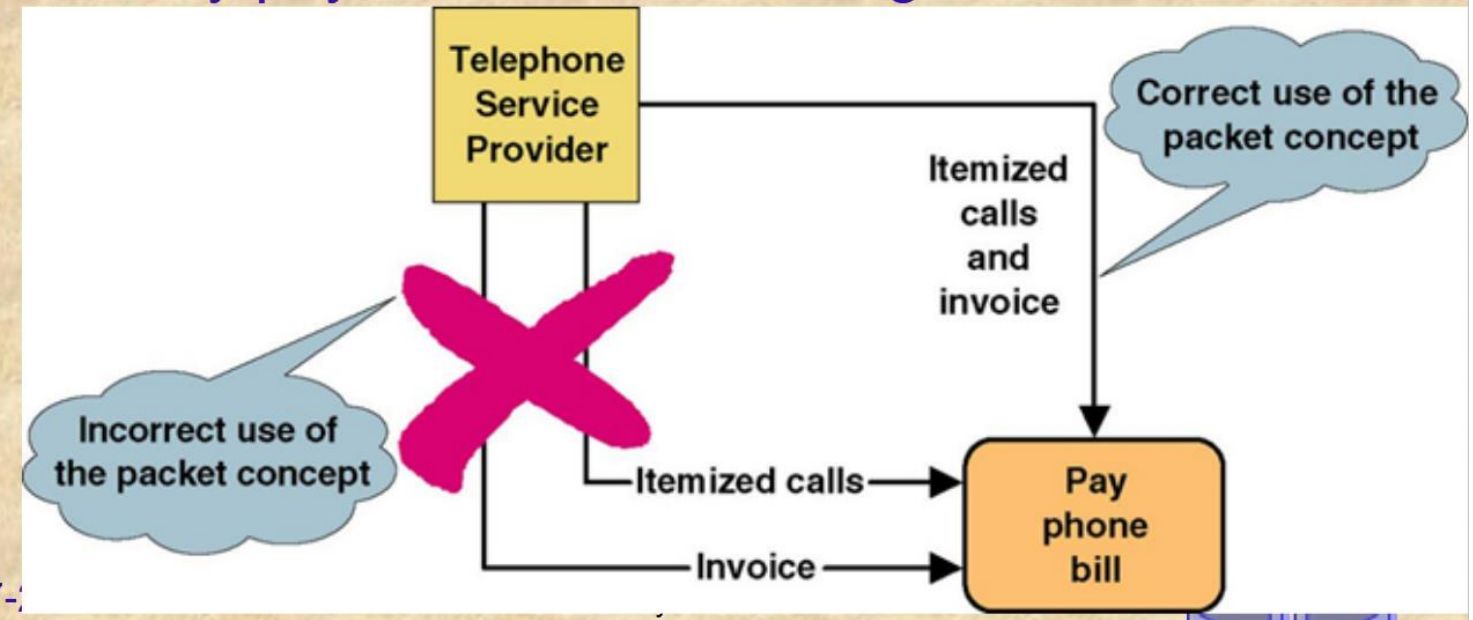
Data Flows & Control Flows

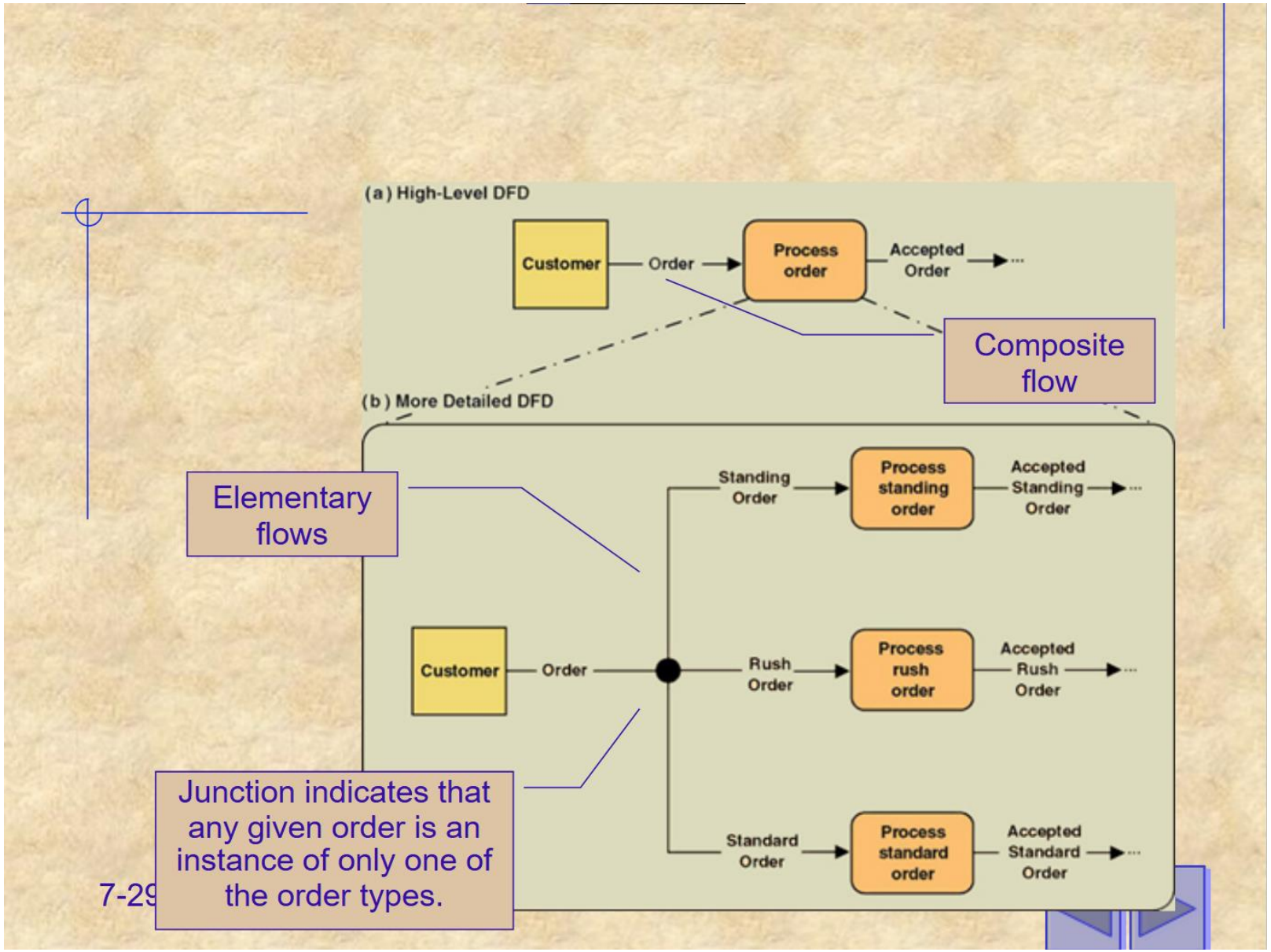




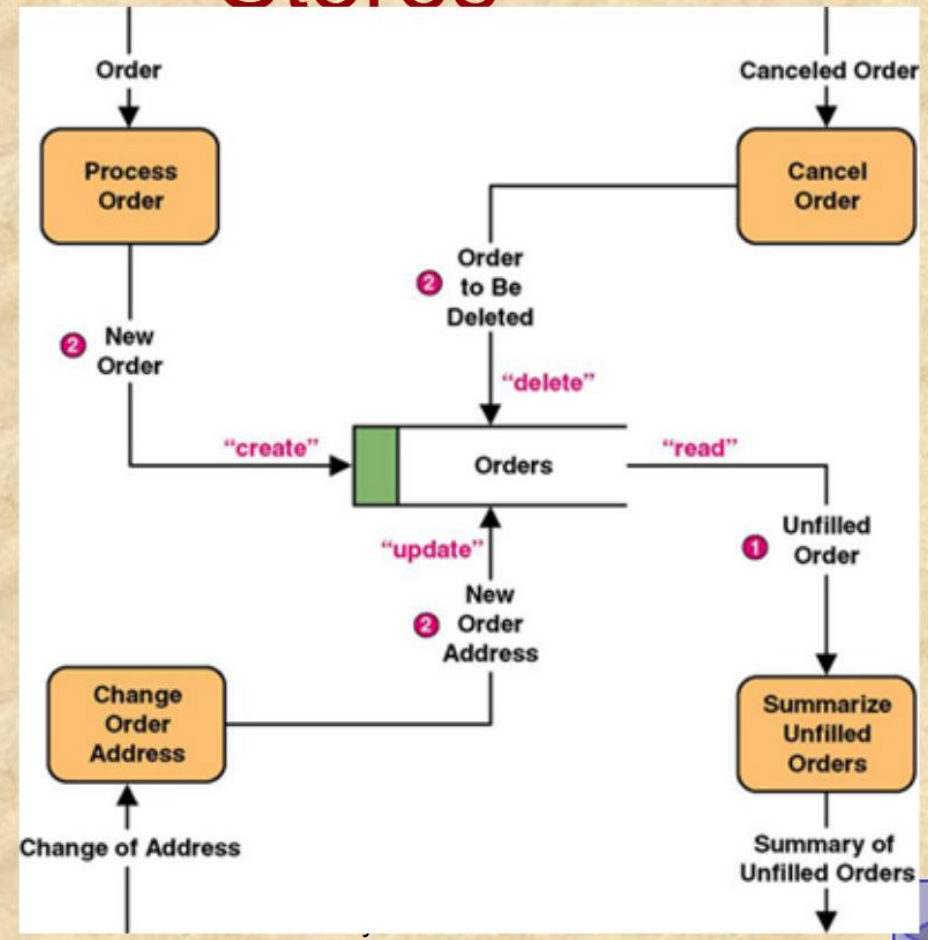
Data Flow Packet Concept

- Data that should travel together should be shown as a single data flow, no matter how many physical documents might be included.



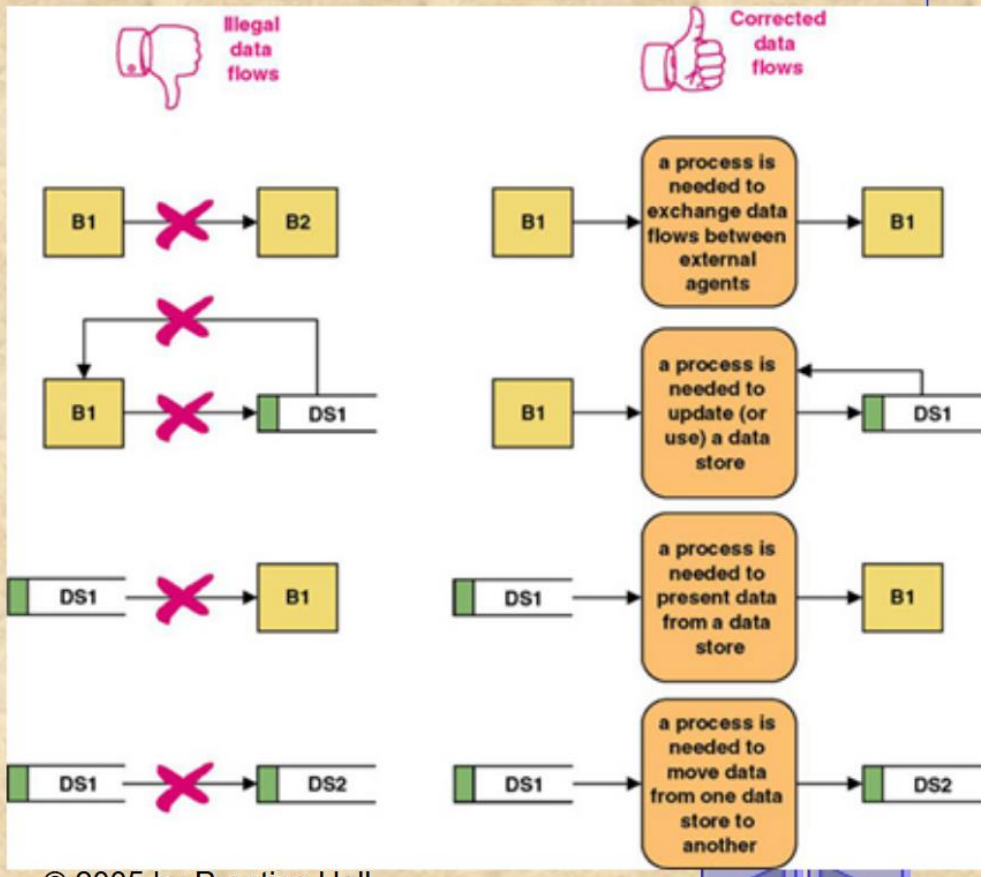

(a) High-Level DFD
Customer
Order
Process order
Accepted Order
Composite flow
(b) More Detailed DFD
Elementary flows
Standing Order
Process standing order
Accepted Standing Order
Customer
Order
Rush Order
Process rush order
Accepted Rush Order
Standard Order
Process standard order
Accepted Standard Order
Junction indicates that any given order is an instance of only one of the order types.

Data Flows to and from Data Stores



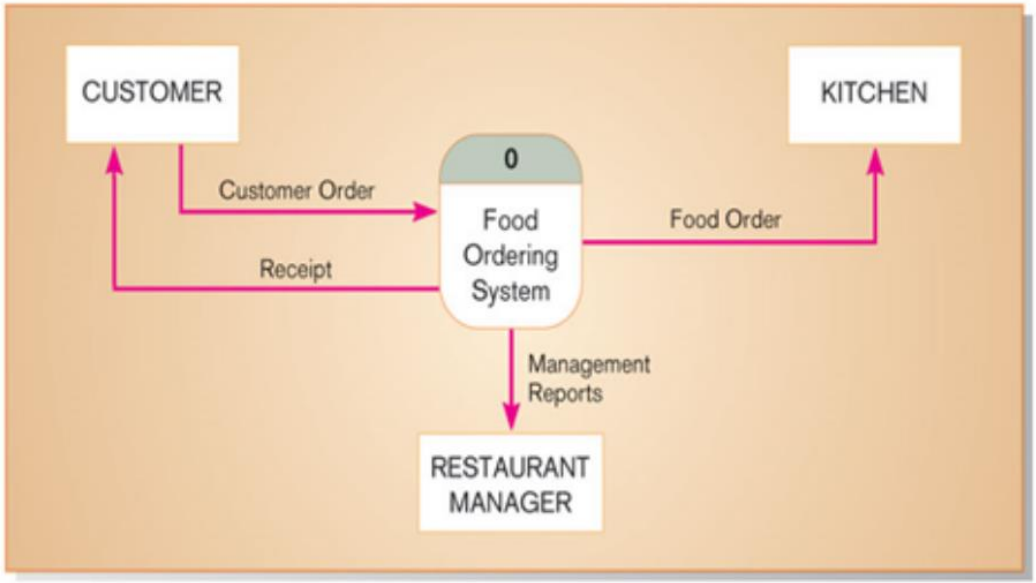
Rules for Data Flows

- A data flow should never go unnamed.
- In logical modeling, data flow names should describe the data flow without describing the implementation
- All data flows must begin and/or end at a process.



Context Diagram

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

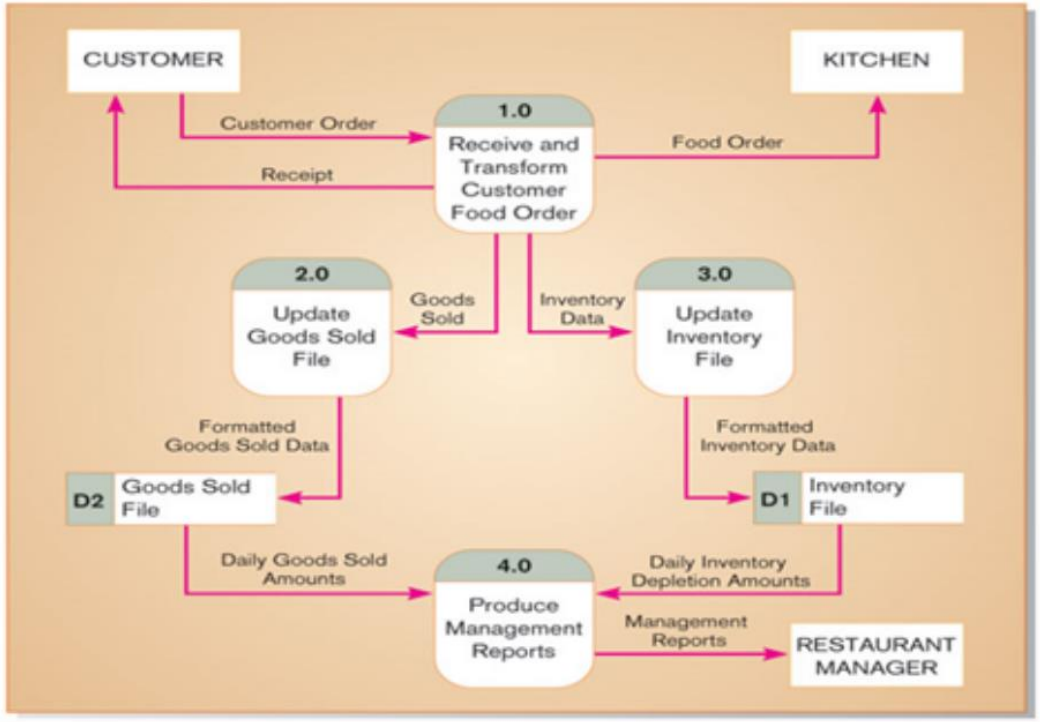


Context diagram shows the system boundaries, external entities that interact with the system, and major information flows between entities and the system.

NOTE: only one process symbol, and no data stores shown.

Level-0 DFD

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

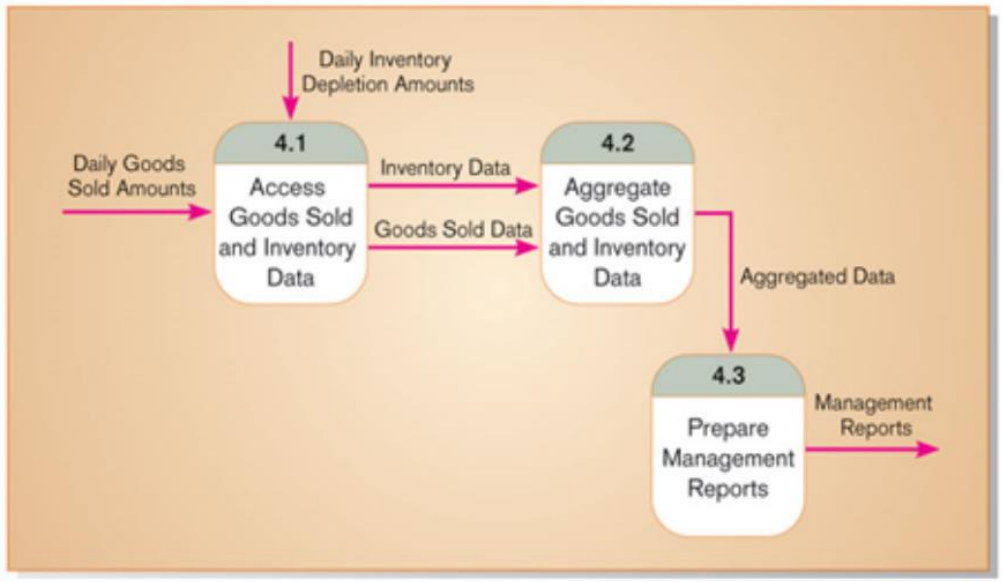


Level-0 DFD shows the system's major processes, data flows, and data stores at a high level of abstraction.

Processes are labeled 1.0, 2.0, etc. These will be decomposed into more primitive (lower-level) DFDs.

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



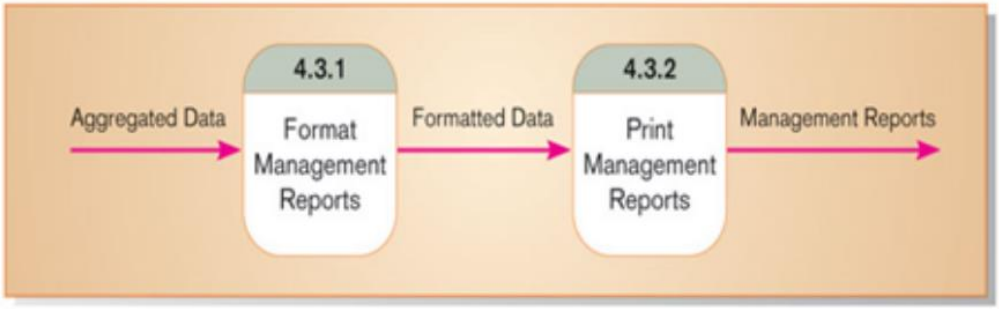
Level-1 DFD shows the sub-processes of one of the processes in the Level-0 DFD.

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

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



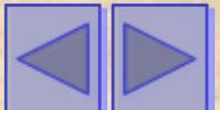
Level-*n* DFD shows the sub-processes of one of the processes in the Level *n-1* DFD.

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.*

DFD Balancing

- The conservation of inputs and outputs to a data flow process when that process is decomposed to a lower level
- 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



Unbalanced DFD

Figure 7-10a An unbalanced set of data flow diagrams - Context diagram

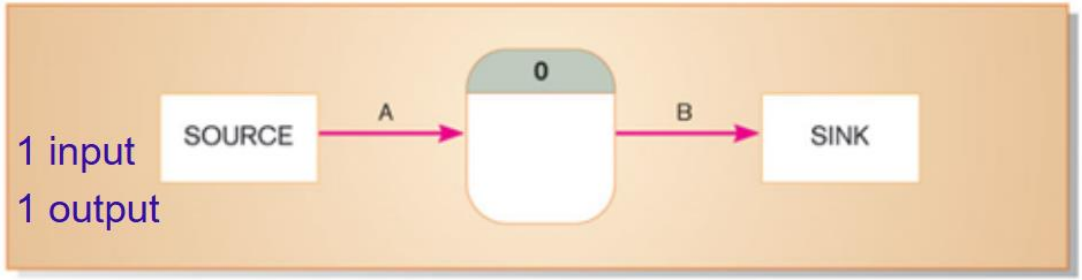
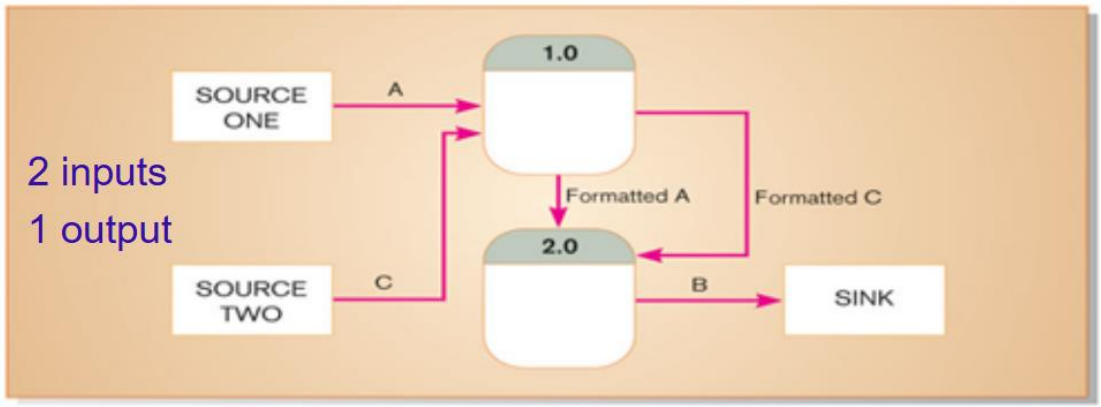


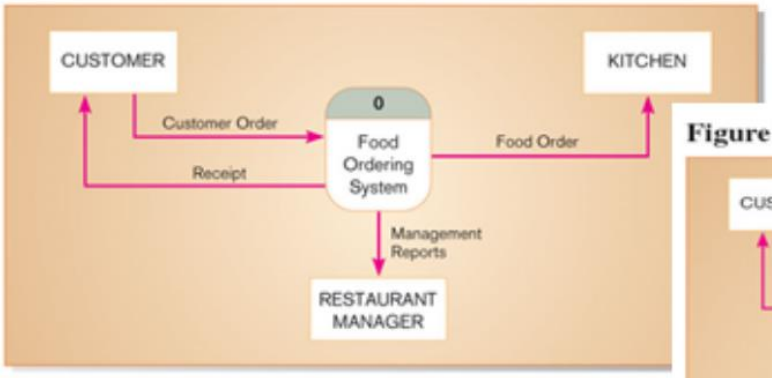
Figure 7-10b An unbalanced set of data flow diagrams - Level-0 diagram



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

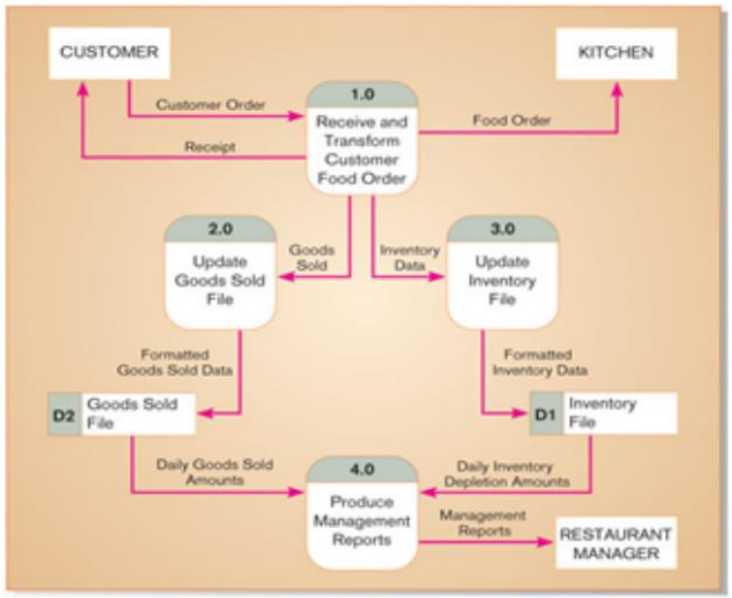
Balanced DFD

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



1 input
2 outputs

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

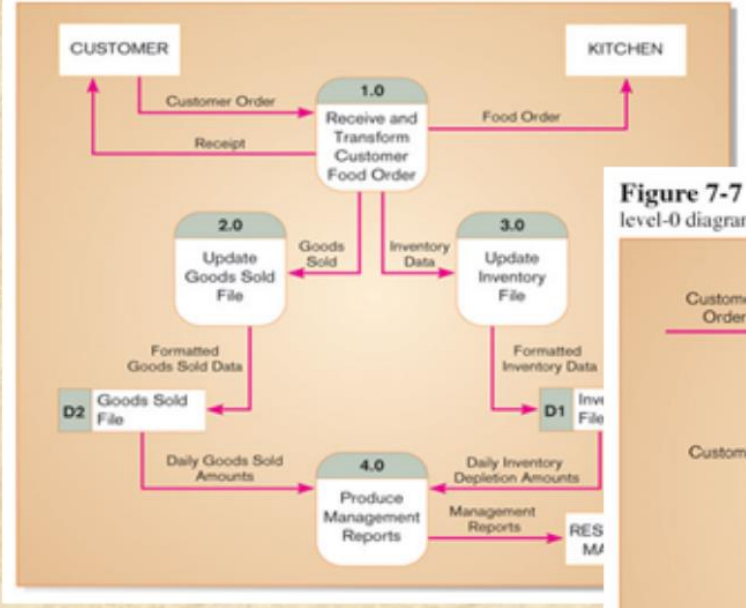


These are balanced because the numbers of inputs and outputs of context diagram process equal the number of inputs and outputs of Level-0 diagram.

Balanced DFD (cont.)

These are balanced because the numbers of inputs and outputs to Process 1.0 of the Level-0 diagram equals the number of inputs and outputs to the Level-1 diagram.

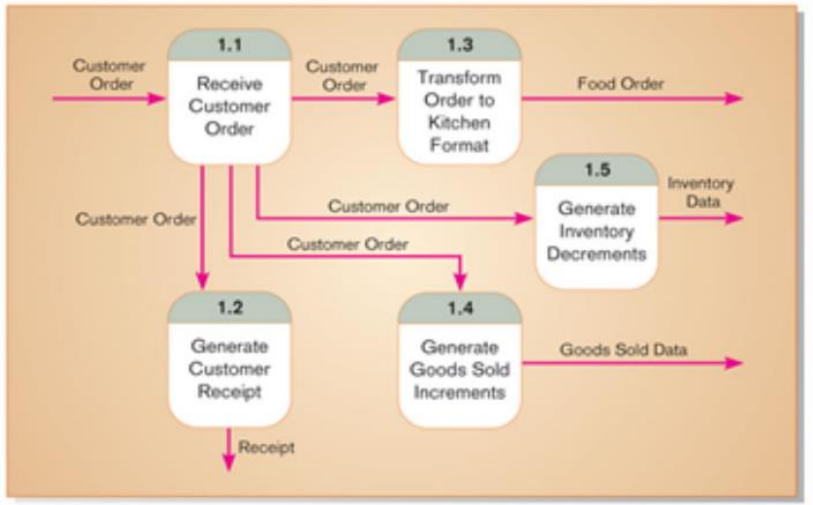
Figure 7-5 Level-0 DFD of Hoosier Burger's food ordering system



1 input

4 outputs

Figure 7-7 Level-1 diagram showing the decomposition of Process 1.0 from the level-0 diagram for Hoosier Burger's food ordering system



Data Flow Splitting

Figure 7-11a Example of data flow splitting - Composite data flow

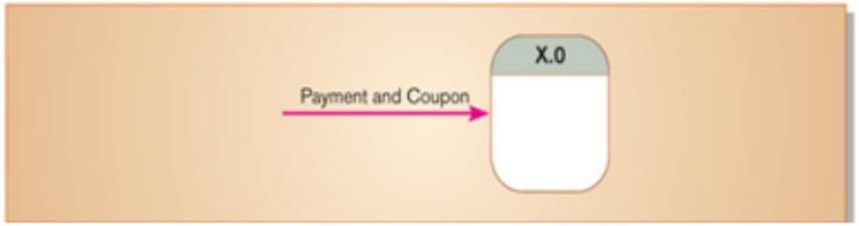
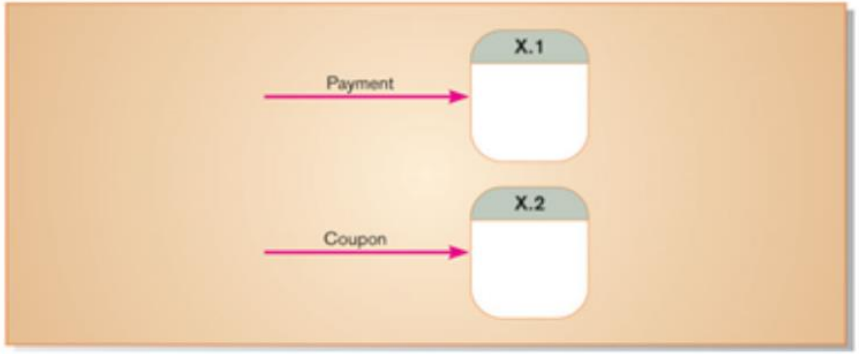


Figure 7-11b Example of data flow splitting - Disaggregated data flows



A composite data flow at a higher level may be split if different parts go to different processes in the lower level DFD.

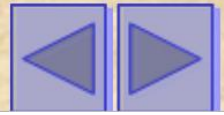
This remains balanced because the same data is involved, but split into two parts.

More DFD Rules

Table 7-3 Advanced Rules Governing Data Flow Diagramming

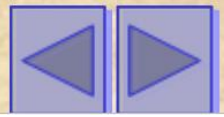
Q.	A composite data flow on one level can be split into component data flows at the next level, but no new data can be added and all data in the composite must be accounted for in one or more subflows.
R.	The inputs to a process must be sufficient to produce the outputs (including data placed in data 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.)



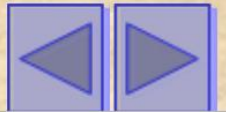
Four Different Types of DFD

- 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.



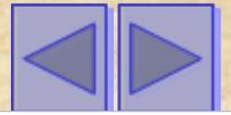
Four Different Types of DFD (cont.)

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



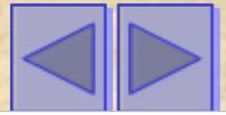
Guidelines for Drawing DFDs

- 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.



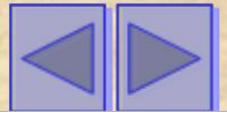
Guidelines for Drawing DFDs (cont.)

- 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
  - When the system user does not care to see any more detail