

# Chapter 7

## Using Data Flow Diagrams



Systems Analysis and Design  
Kendall & Kendall  
Sixth Edition

# Major Topics

---

- Data flow diagram symbols
- Data flow diagram levels
- Creating data flow diagrams
- Physical and logical data flow diagrams
- Partitioning
- Event driven modeling
- Use case and data flow diagrams

# Data Flow Diagrams

---

- DFDs are one of the main methods available for analyzing data-oriented systems.
- DFDs emphasize the logic underlying the system.
- The systems analysts can put together a graphical representation of data movement through the organization.

# Advantages of the Data Flow Diagram Approach

---

Four advantages over narrative explanations of data movement:

- Freedom from committing to the technical implementation too early.
- Understanding of the interrelationships of systems and subsystems.
- Communicating current system knowledge to users.
- Analysis of the proposed system.

# Basic Symbols

---






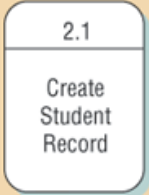


Four basic symbols are:

- A double square for an external entity--a source or destination of data.
- An arrow for movement of data from one point to another.
- A rectangle with rounded corners for the occurrence of transforming process.
- An open-ended rectangle for a data store.

# Basic Symbols

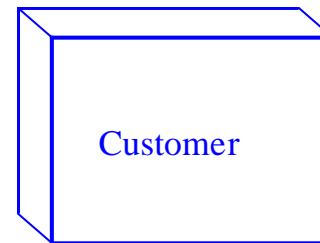
**Figure 7.1**

The four basic symbols used in data flow diagrams, their meanings, and examples.

Symbol	Meaning	Example
	Entity	
	Data Flow	
	Process	
	Data Store	

# External Entities

- Represent people or organizations outside of the system being studied
- Shows the initial source and final recipient of data and information
- Should be named with a noun, describing that entity



# External Entities (Continued)

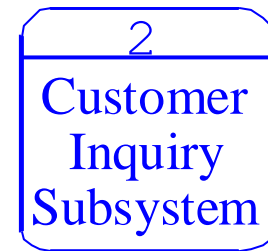
---

- External entities may be:
  - A person, such as CUSTOMER or STUDENT.
  - A company or organization, such as BANK or SUPPLIER.
  - Another department within the company, such as ORDER FULFILLMENT.
  - Another system or subsystem, such as the INVENTORY CONTROL SYSTEM.



# Processes

- Represent either:
  - A whole system
  - A subsystem
  - Work being done, an activity
- Names should be in the form verb-adjective-noun
  - The exception is a process that represents an entire system or subsystem.



# Data Stores

- Name with a noun, describing the data
- Data stores are usually given a unique reference number, such as D1, D2, D3.
- Include any data stored, such as:
  - A computer file or database.
  - A transaction file .
  - A set of tables .
  - A manual file of records.

D1	Customer Master
----	--------------------

# Data Flow



- Data flow shows the data about a person, place, or thing that moves through the system.
- Names should be a noun that describes the data moving through the system.
- Arrowhead indicates the flow direction.
- Use double headed-arrows only when a process is reading data and updating the data on the same table or file.

# Developing Data Flow Diagrams

Use the following guidelines:

- Create the context level diagram, including all external entities and the major data flow to or from them.
- Create Diagram 0 by analyzing the major activities within the context process.
  - Include the external entities and major data stores.
- Create a child diagram for each complex process on Diagram 0.

# Creating Data Flow Diagrams

---

Detailed data flow diagrams may be developed by:

- Making a list of business activities.
- Analyzing what happens to an input data flow from an external entity.
- Analyzing what is necessary to create an output data flow to an external entity.

# Creating Data Flow Diagrams

---

Detailed data flow diagrams may be developed by (continue):

- Examining the data flow to or from a data store.
- Analyzing a well-defined process for data requirements and the nature of the information produced.
- Noting and investigating unclear areas.

# Data Flow Diagram Levels

---

- Data flow diagrams are built in layers.
- The top level is the Context level.
- Each process may explode to a lower level.
- The lower level diagram number is the same as the parent process number.
- Processes that do not create a child diagram are called primitive.

# Context-Level Data Flow Diagram

- It contains only one process, representing the entire system.
- The process is given the number zero.
- All external entities are shown on the context diagram as well as major data flow to and from them.
- The diagram does not contain any data stores.



# Diagram 0

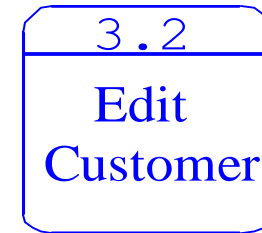
---

- Diagram 0 is the explosion of the context level diagram.
- It should include up to 7 or 9 processes.
  - Any more will result in a cluttered diagram.
- Processes are numbered with an integer.
- The major data stores and all external entities are included on Diagram 0.

# Child Diagrams

---

- Each process on diagram zero may be exploded to create a child diagram.
- Each process on a lower-level diagram may be exploded to create another child diagram.
- These diagrams found below Diagram 0 are given the same number as the parent process.
  - Process 3 would explode to Diagram 3.



## Child Diagrams (Continued)

- Each process is numbered with the parent diagram number, a period, and a unique child diagram number.
- Examples are:
  - 3.2 on Diagram 3, the child of process 3.
  - 5.2.7 on Diagram 5.2, child of process 5.2.
  - On Diagram 3, the processes would be numbered 3.1, 3.2, 3.3 and so on.

# Child Diagrams (Continued)

---

- External entities are usually not shown on the child diagrams below Diagram 0.
- If the parent process has data flow connecting to a data store, the child diagram may include the data store as well.

# Child Diagrams (Continued)

---

- A lower-level diagram may contain data stores not shown on the parent process, such as:
  - A file containing a table of information (such as a tax table).
  - A file linking two processes on the child diagram.
- Minor data flow, such as an error line, may be included on a child diagram.

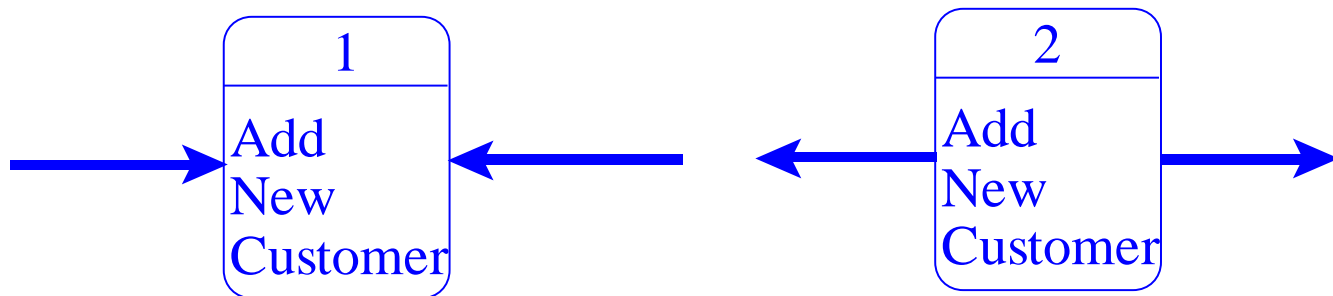
# Child Diagrams (Continued)

---

- An interface data flow is data that are input or output from a child diagram that matches the parent diagram data flow.
- Processes that do not create a child diagram are called primitive processes.
- Logic is written for these processes.

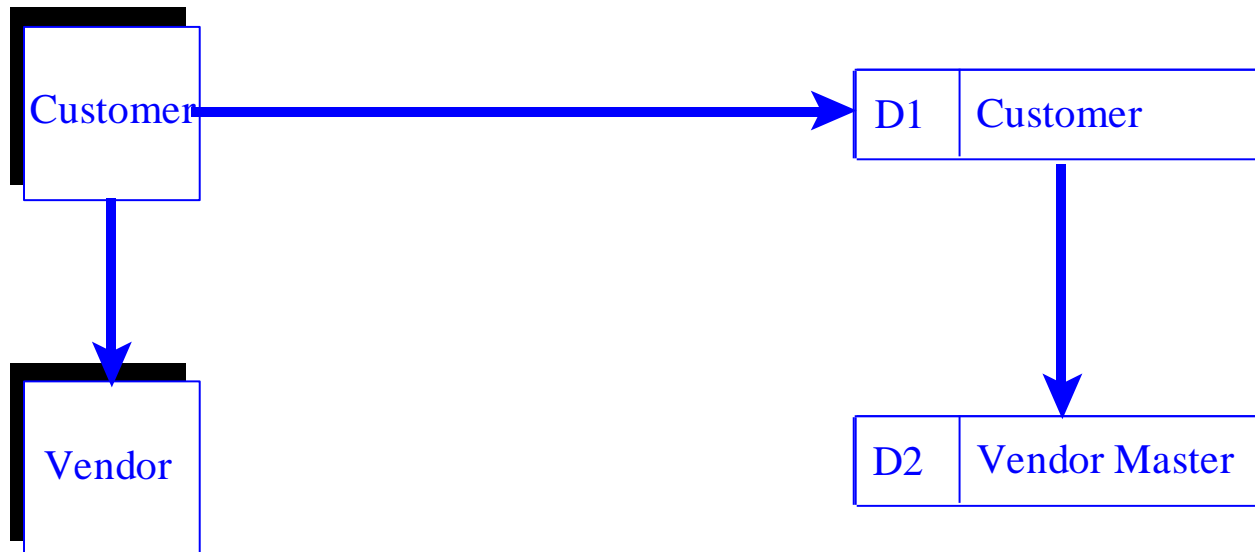
# Data Flow Diagram Errors

- The following conditions are errors that occur when drawing a data flow diagram:
- A process with only input data flow or only output data flow from it.



# Data Flow Diagram Errors (Continued)

- Data stores or external entities are connected directly to each other, in any combination.





# Data Flow Diagram Errors (Continued)

- Incorrectly labeling data flow or objects
  - Examples are:
    - Labels omitted from data flow or objects.
    - Data flow labeled with a verb.
    - Processes labeled with a noun.
- Too many processes on a data flow diagram.
  - Nine is the suggested maximum.

# Data Flow Diagram Errors (Continued)

---

- Omitting data flow from the diagram
- Unbalanced decomposition between a parent process and a child diagram
  - The data flow in and out of a parent process must be present on the child diagram.

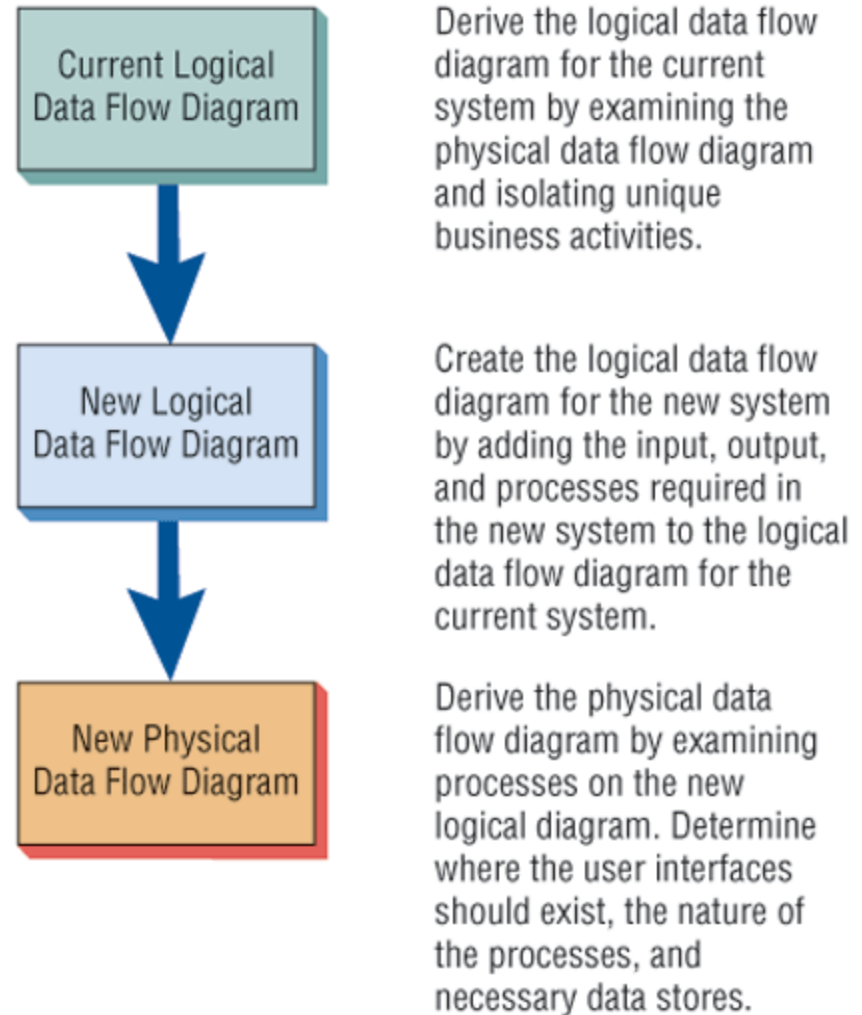
# Logical Data Flow Diagrams

---

- Logical data flow diagrams show how the business operates.
- They have processes that would exist regardless of the type of system implemented.

# Data Flow Diagram Progression

**Figure 7.8** The progression of models from logical to physical.



# Logical Data Flow Diagrams

## Advantages

---

Advantages of logical DFDs are:

- Better communication with users.
- More stable systems, since the design is based on a business framework.
- Increased understanding of the business by analysts.
- The system will have increased flexibility and be easier to maintain.
- Elimination of redundancy.

# Physical Data Flow Diagrams

---

Physical data flow diagrams show how the system operates or how the new system will be implemented.

- Physical data flow diagrams include:
  - Clarifying which processes are manual and which are automated.
  - Describing processes in greater detail.
  - Sequencing processes in the order they must be executed.

# Physical Data Flow Diagrams

---

Physical data flow diagrams include (continued):

- Temporary data stores and transaction files.
- Specifying actual document and file names.
- Controls to ensure accuracy and completeness.

# CRUD

---

- Physical data flow diagrams include processes for adding, reading, changing, and deleting records.
- CRUD is an acronym for Create, Read, Update, Delete.
- A CRUD matrix shows which programs or processes add, read, update, or delete master file records.



# Transaction Files

---

- Master or transaction database tables or files are used to link all processes that operate at different times.
- They are required to store the data from the process that creates the data to the process that uses the data.

# Partitioning

---

- Partitioning is the process of analyzing a data flow diagram and deriving a series of manual procedures and computer programs.
- A dashed line is drawn around a group of processes that are included in each computer program or manual procedure.

# Reasons for Partitioning

---

- The reasons for partitioning a data flow diagram into separate computer programs are:
  - Different user groups should have different programs.
  - Processes that execute at different times must be in separate programs.
  - Processes may be separated into different programs for security.

# Reasons for Partitioning (Continued)

---

- Similar tasks may be included in the same program.
- Several batch processes may be included in the same program for efficiency.
- Several processes may be included in the same program or job stream for consistency of data.

# Partitioning Web Sites

---

Web sites are partitioned into pages.

- Improves speed of processing
- Easier Web page maintenance
- Different pages when reading different data
- Partitioned for security, separating pages using a secure connection from those that do not

# Communicating Using Data Flow Diagrams

---

Data flow diagrams can be used for several different purposes:

- Unexploded data flow diagrams are useful to identify information requirements.
- Meaningful labels should be used for good communication.