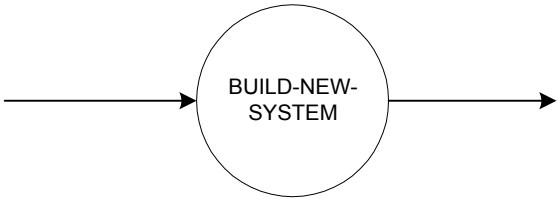
Chapter 5

Data Flow Diagrams

A. Introduction

Data Flow diagrams are the primary graphic tools to analyze a system. The systems analyst uses DFDs to show what happens to data items as they flow through a system.

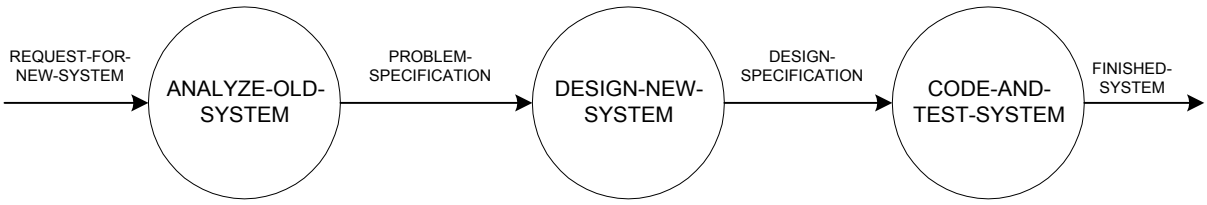
A data flow diagram is a model of the processes that transform the data.



On a data flow diagram:

- The arrows indicate that the pieces of data the name of which is put above the arrows move through the system
- The circles show the process done to the data

A central process can be broken down into smaller processes. By partitioning a function into component-functions we can understand better what is going on in the system.



A system can be partitioned into tasks, subtasks, and sub-subtasts. If the system is broken down sufficiently in the DFDs, the activities that follow will be easier

B. Elements of the Data Flow Diagram

The Terminator Symbol: rectangle containing terminator name

- The terminator or **external entity** is a person or organization, outside the boundaries of the system, who supplies to the system (**source**) or receives data from the system (**sink**).

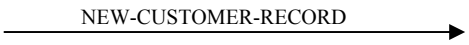
The Data Store Symbol: a rectangle open in one end

- A data store is a repository (disk file, tape, file cabinet, etc.) or resting-place for data. It stays there until needed by some part of a system
- The arrows, or data flows, entering and exiting the data stores show the directions in which the data travel.
- An arrow ending a data store indicates an update (add, change, or delete) and an arrow starting a data store indicates a read.

The Process Symbol: a circle with process-name inside

- A process is a task or a function that must be completed
- The process indicates a transformation of the data: the data leaving a process is different from the data that entered it.
- A **black box** is a task for which we know the inputs and outputs and the general function, but does not know how the function is accomplished.
- A black box behaves predictably: given a particular input, it always produces the same output

The Data Flow Symbol: an arrow labeled with the name of the data flow.



- Data flows serve as communication channels between processes
- Each data flow must have an unique name, and the name must be specific enough to differentiate all the data flows in a system
- By convention data flow names are always singular. We assume that a single piece of data is moving through the system. No reiteration, no flags on a DFD.
- All data flows must either begin or end at a process symbol. Data flows cannot connect two terminators, two data stores, or a terminator and a data store.

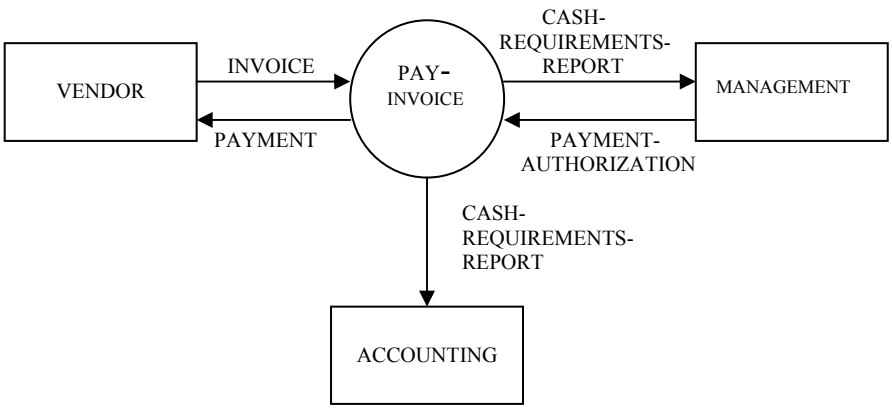
C. Context Data Flow Diagram

A context data flow diagram is the highest level diagram, the most abstract version of the system.
A context data flow diagram only shows the external view: inputs, outputs, terminator, and the general function of the system.

Terminators and the Context DFD

The Context data flow diagram is the only diagram that shows terminators. At this high level, terminators are needed to show who gives and who gets which data.

Whenever possible, the prime source is shown on the left and the prime sinks on the right.



Data Stores in the Context DFD

- At the top level, data stores are not shown because they are generally internal to a system

Process name in the Context DFD

- At the top level, the process should summarize all the functions that the system performs.
- The process name is often a noun or noun phrase, without a verb (at this highest level, the verb can be omitted)

Data-flow names in the Context DFD

- Data flow names are in singular even though there may be many items going through.
- The same data flow item can appear twice or more times on the context DFD because they are multiple copies of the same item flowing to different directions.
- In a context DFD, multiple names may appear on the same arrow.

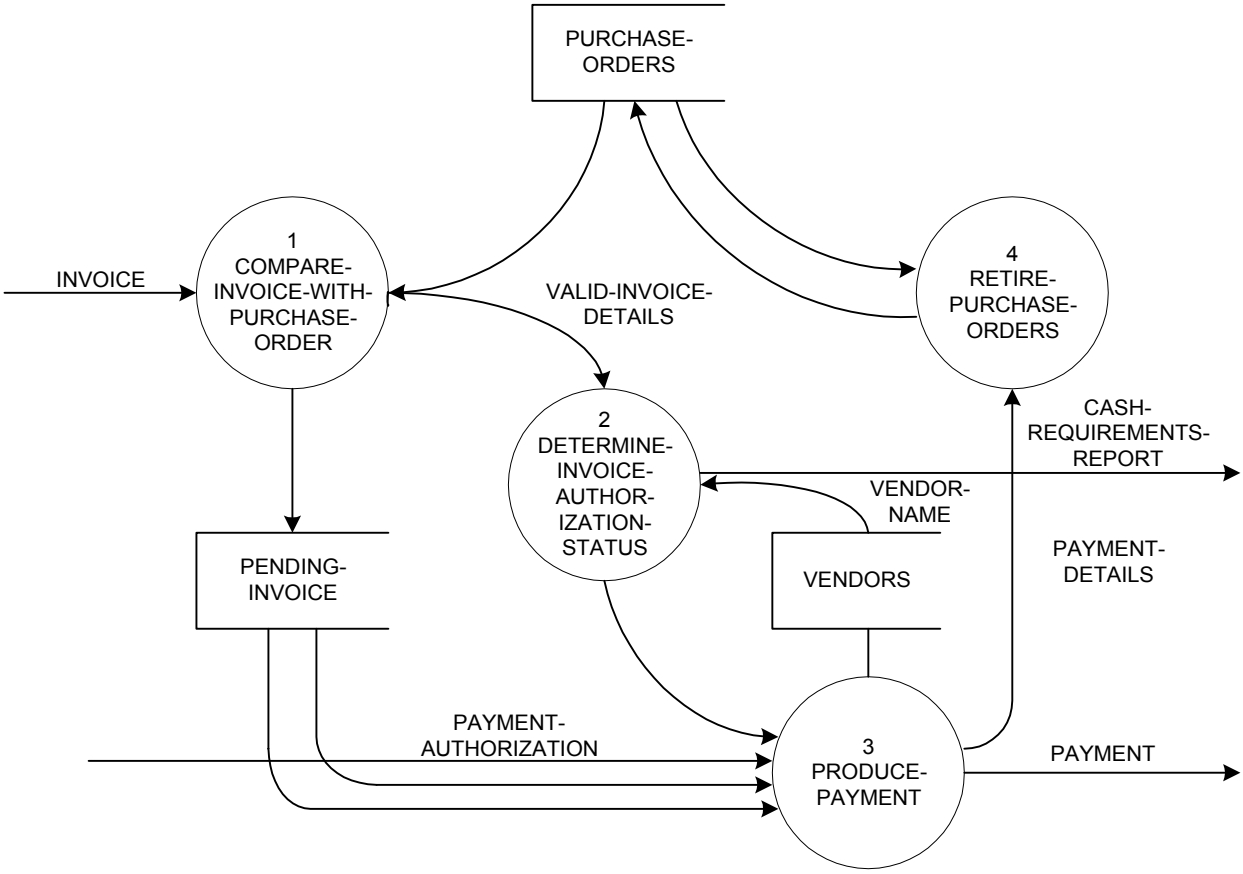
D. Overview Data Flow Diagram

- This is the level next to the context data flow diagram. The overview data flow diagram is a subsystem of the context data flow diagram - this process of breaking down system to subsystems is called **leveling**.
- The overview data flow diagram shows the major data stores, data flows, and processes of the system.

Data Stores in the Overview DFD

- We need a data store:
    1. Whenever data must be held somewhere for some time
    2. When records must be accessed in an order different from the order in which they entered the system

- A data store is set at the highest level of the overview diagram when two or more processes access it. All lower level diagrams will use the established data store (rule 1)
- We must disclose all references to the data store established at the highest level (rule 2)



- Sometimes a data store may be the only connection between two processes. This happens often on overview diagrams whenever there is a time delay between processes

Processes and the Overview DFD

- **Partitioning** is dividing units of work among different processes on a DFD. It is often difficult to decide exactly how the work should be partitioned. Partitioning is a skill acquired only after much practice.
- We may have to make many sketches before creating a satisfactory DFD. In general the best partitioning solution is the one that minimizes the number of data flows between processes and thus simplifies the entire diagram.
- We should try whenever possible to limit a given DFD to seven processes. We may have fewer than seven processes on a DFD. The number of processes should illustrate best the division of labor.

- Three guidelines:
    1. When partitioning an overview diagram, we do not show processes for trivial functions, such as editing or formatting. These are left to lower-level diagrams.
    2. We should never combine two different time frames into a single process on the overview diagram
    3. For each major input to the system we should start with a single process.

Data Flows and the Overview DFD

- Data flow names on the overview diagram are unique (and generally long)., one name per data flow, with the arrowhead showing the direction of the data movement. They do not indicate a reiteration, nor are they used as flags.
- A data flow name may be omitted on an arrow in or out of a data store if the entire file record or even most of it is used.

Balancing

- A child diagram must have the same inputs and outputs as the parent process. This is the concept called balancing.
- Balancing refers only to the inputs and outputs to the parent process, and the inputs and outputs leaving the edge of the page on the child diagram.
- We cannot build the outputs without the appropriate inputs. In order to produce the desired output, a process must be supplied with the necessary inputs. This is the concept of **data conservation**.
- When two arrows come together under one name, there is **convergence**. This merging shows that identical copies of the same data can come from two sources.
- The opposite of a convergence is a **divergence**, a point where a data flow arrow divides, and sending identical copies of the data to two or more destinations.
- We should use convergence or divergence whenever possible rather than to show two separate copies of the sale data flow. The exception to this is when using convergence or divergence results in an excess of crossed data flows on the diagram.

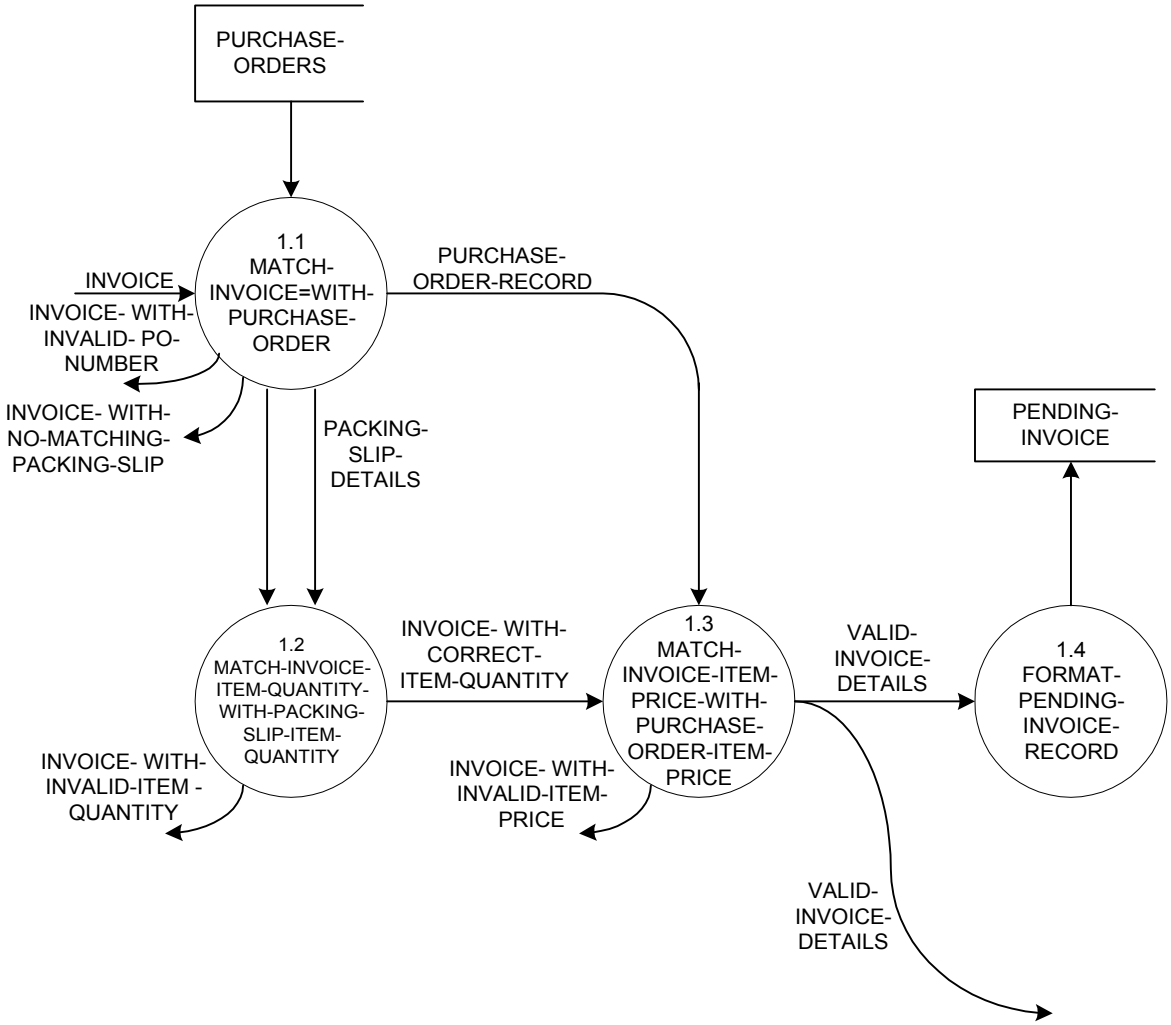
E. Lower-Level Data Flow Diagrams

- These diagrams partition the individual processes shown on the overview DFD. Each is the child of a specific process shown in the overview DFD.

- The numbering scheme of these diagrams is based on the numbers of their parent process. For instance, the diagram partitioning process 1 on the overview DFD will have its own processes numbered 1.1, 1.2, 1.3, 1.4, …, 1.7.
- If we were to partition process 1.1 further, the lower-level diagram would have its processes numbered 1.1.1, 1.1.2, 1.1.3, …, 1.1.7

Parallel Decomposition

- Parallel Decomposition happens when an arrow on the parent becomes two or more arrows on the child. The single data flow is broken down into its components or alternative versions.
- Parallel decomposition is necessary when components originate from or go to different places.



- Parallel decomposition also occurs when the parent data flow takes the form of a series of choice on the child diagram.
- Parallel decomposition makes the DFD harder to read however. In order to verify that the components are equal to the original, we must refer to the data dictionary. The reference to the data dictionary is annoying. Therefore, we should use parallel decomposition sparingly.

Tramp data

- Tramp data are data items that are shuffled through one or more processes unnecessarily. We avoid this as much as possible. We "starve" the processes, delivering to each only that data it must have. Every time a piece of data enters a process, it runs the risk of being altered accidentally, or it may cause some unexpected errors.

Siblings

- Siblings are processes that have the same parents and , together are the decomposition of that parent.

How Much Partitioning is Necessary?

- Not enough partitioning may be detrimental to a system model. We might need to partition further if there is at least a process that:
    1. Seems to perform more than one distinct function.
    2. Is difficult to name; it may be that the process is trying to handle multiple functions
    3. Would be easier to understand if it were broken into smaller tasks.
    4. Has either many inputs or many outputs or both, which is asign that there might be too much going on inside the process.

- On the lowest-level diagram, the process that cannot be partitioned further is called a **functional primitive.**

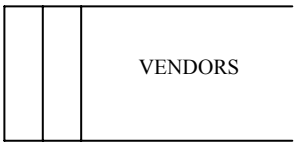
F. Method of Constructing Data Flow Diagrams

1. Identify all terminators and data flows that connect them to the system.
2. Use the information in step 1 to construct the context DFD.
3. Use the following step to do the partitioning:
    - Sketch the first draft: identify the processes and data flows. Make sure that the proper inputs are supplied to produce the necessary outputs for each process
    - Create as many drafts as necessary until the results seem right
    - Uncover possible errors
    - Draw up the last version of each diagram neatly
4. Put the finished versions of all diagrams in order. Make copies and conduct a walkthrough with team members. Repeat step 3 if needed.
5. Conduct a walkthrough with users. Return to step 3 as needed.
6. Produce a final draft of each diagram.

- As we work on our DFDs, we simultaneously create entries for the data dictionary and write the process specifications.
- Draw the final versions on 8 ½ X 11 paper to make them fit in binders.
- Never use an off-page connector.

- Avoid crossed lines. When needed, use a hoop convention or a broken line
- A data flow should never enter or exit the right side (the open end) of a data store.
- Always write the name on a data flow horizontally, even if the data flow slants off in some other direction.
- Reflect the timing of the processes by arranging them from left to right and from top to bottom.
- When there are several crossed lines, we may need multiple copies of the data store. Adding another copy of the data store may eliminate the crossed lines.
- Indicate multiple copies of a data store by using a series of vertical lines down the left side of the symbol.

2 copies of purchase-orders



3 copies of Vendors

G. Signs of errors in Data Flow Diagram

- The data flows strangle the diagram, especially lower-levels ones, filling every inch: Try different partitioning, move some bubbles to another level to minimize interfaces.
- Some area streams for a flag: a single function might have been divided in two or more processes. Reunite the wrongly divided processes.
- An output does not correspond to appropriate inputs (a "**miracle**") or inputs that are never used.
- A bubble that has no outputs (a "**black hole**") is not accomplishing any useful function
- Data flows or processes for which we have trouble thinking of descriptive names.
- There is a wide discrepancy in the degree of leveling in the different DFDs in a set, e.g. one function has three levels and another has nine levels of partition.
- The DFD shows data composition, data access methods, decisions, loops, descriptions of the work inside a process. There is redundancy: the data dictionary handles the first two items and the rest appear in the process descriptions.