Overview: Data flow diagrams model how data flow through an information system, the relationships among the data flows, and how data come to be stored at specific locations. DFDs show the processes that change or transform the data.

Graphic tool

What to do and what not to do: balancing and decomposition.

Four types of DFDs: current physical, current logical, new logical, and new physical.

Please be sure to review the UML examples.

DFD – modeling a system’s process. Previously worked on requirements determination.

Deliverables

1. Context data flow diagram (DFD)
2. DFDs of current physical system (not too much detail)
3. DFDs of current logical system
4. DFDs of new logical system
5. Thorough descriptions of each DFD component

DFDs:

- Data flow can represent anything: data at rest (data store) and data in motion (data flow).
- Process: is a work or action performed on the data so they’re transformed, stored, distributed.
- Source: origin of the data
- Sink: destination of the data.
- Sources & sinks are referred to as external entities because they go outside the system.
- Don’t need to consider the following:
  i. Interactions that occur between sources and sinks
  ii. What a source or sink does with the data or how it operates (a black box)
  iii. How a control or redesign a source or sink since the info system deals with the data as they are (what goes in and out of the box)
  iv. How to provide sources and sinks direct access to stored data because they cannot manipulate the data: the system must receive or distribute data between the system and its environment

Context Diagram

Highest level is the context diagram: notice contains only one process, no data stores, some flows, some sinks. The single process is labeled “0” and represents the entire system.

Next think about which processes are represented by the single process in the context diagram.

Break down into separate processes (Level-n diagrams); each process gets its own number.
Differences between sources/sinks & processes.
Above is an example of an improperly drawn DFD showing a process as a source/sink.
A DFD showing the proper use of a process.

DFD Rules:
Process:
- No process can have only outputs. It is making data from nothing. If an object has only outputs, then it must be a source.
- No process can have only inputs (a black hole). If an object has only inputs, then it must be a sink.
- A process has a verb phrase label.

Data Store:
- Data cannot move directly from one store to another data store. Data must be moved by a process.
- 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.
- Data cannot move directly to an outside sink from a data store. Data must be moved by a process.
- Data store has a noun phrase label.

Source/Sink:
- Data cannot move directly from a source to a sink.
- A source/sink has a noun phrase label.

Data Flow:
- 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 since these happen at different times.
- A fork in a data flow means that exactly the same data go 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).

- A join 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 to a common location.
- A data flow cannot go back directly 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.
- A data flow to a data store means update (delete or change)
- A data flow from a data store means retrieve or use.
- 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.

Advanced Rules governing DFD:

- 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 sub-flows.
- 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, either to another process or to a data store outside the process or on a more detailed DFD showing a decomposition of that process.
- 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?”)
- 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.

[adapted from Celko, 1987]

Decomposition:

From single system to component processes is called functional decomposition. From the context level diagram, create Level-0 and continue decomposition for each process (numbered sequentially, level-1, level-2, etc., called level-n diagram).

Balancing DFDS:

- The sub-processes must be balanced; e.g., if Process 1.0 appears in level-0 must have the same inputs and outputs when decomposed into a level-1 diagram.
- The lowest level of the level-n diagram is a “primitive.” [More later.]
Context Diagram of a Digital Library search system

Level-0 DFD of a Digital Library search system
Level-1 DFD showing the decomposition of Process 1.0 from the Level-0 Diagram

1.1 Receive user query

1.2 Generate user receipt

1.3 Transform query to request format

1.4 Generate user visits increments (update)

1.5 Generate access-log update

user request

user request

user request

user visits data

Level-1 diagram showing the decomposition of Process 4.0 from the Level-0 diagram

4.1 Access user queries and transaction log data

4.2 Aggregate user queries and transaction log data

4.3 Prepare management reports

Daily transaction log updates

Aggregated data

Daily user requests

Management reports
Level-2 diagram showing the decomposition of Process 4.3 from the Level-1 diagram for Process 4.0

DFD Consistency:

Consistency refers to whether or not the system shown at one level of a nested set of DFDs is compatible with the depictions of the system shown at other levels, e.g., a level-1 diagram with no level-0 diagram.

Timing:
Notice that DFDs do not indicate when something happens. Time is shown using a state-transition diagram [discussed later].

Primitive DFDs:
- When to stop decomposing? Some rules:
  - When you’ve reduced each process to a single decision or calculation or to a single database operation, such as retrieve, update, create, delete, or read.
  - When each data stores represents data about a single entity, such as a customer, employee, product, or order.
  - When the system user doesn’t care to see more detail, or when you and other analysis have documented sufficient detail to do subsequent systems development tasks.
  - When every data flow doesn’t need to be split further to show that different data are handled in different ways.
  - When you believe that you have shown each business form or transaction, computer online display, and report as a single data flow (this often means that each system display and report title corresponds to the name of an individual data flow).
  - When you believe there is a separate process for each choice on all lowest-level menu options for the system.

Next: Logic modeling.