Chapter 2 - Structuring System Requirements: Process Modeling

Process Modeling:

Process modeling graphically represents the processes that capture, manipulate, store, and distribute data between a system and its environment and among system components. It utilizes the information gathered during requirements determination. Main focus is to model the processes and data structures. It is a formal way of representing how a business system operates. It also illustrates the activities that are performed and how data moves among them.

A *process model* is a formal way of representing how a business operates. It is a core diagram in structured analysis and design. It shows the flow of information through a system. Each process transforms inputs into outputs.

Process models are based on behavior and actions. Example: Data flow diagram

Data Flow Diagram (DFD)

A DFD is a pictorial representation of the movement of data between external entities and the processes and data stores within a system. It is a common technique for creating process models. A DFD shows how data moves through an information system but does not show program logic or processing steps. DFDs can also be used for the visualization of data processing (structured design).

Difference between flowcharts & DFD

The program flowchart describes boxes that describe computations, decisions, interactions & loops. It is an important to keep in mind that data flow diagrams are not program flowcharts and should not include control elements. DFDs depict logical data flow independent of technology but flowcharts depict details of physical systems.

A good DFD should

- ✓ have no data flows that split up into a number of other data flows
- ✓ have no crossing lines
- ✓ not include flowchart loops of control elements
- ✓ not include data flows that act as signals to activate processes

Advantages of data flow diagrams

- ✓ It gives further understanding of the interestedness of the system and sub-systems
- ✓ It is useful from communicating current system knowledge to the user
- ✓ Used as part of the system documentation files
- ✓ Data flow diagram helps to substantiate the logic underlining the dataflow of the organization
- ✓ It gives the summary of the system
- ✓ DFD is very easy to follow errors and it is also useful for quick reference to the development team for locating and controlling errors

Disadvantages of data flow diagram

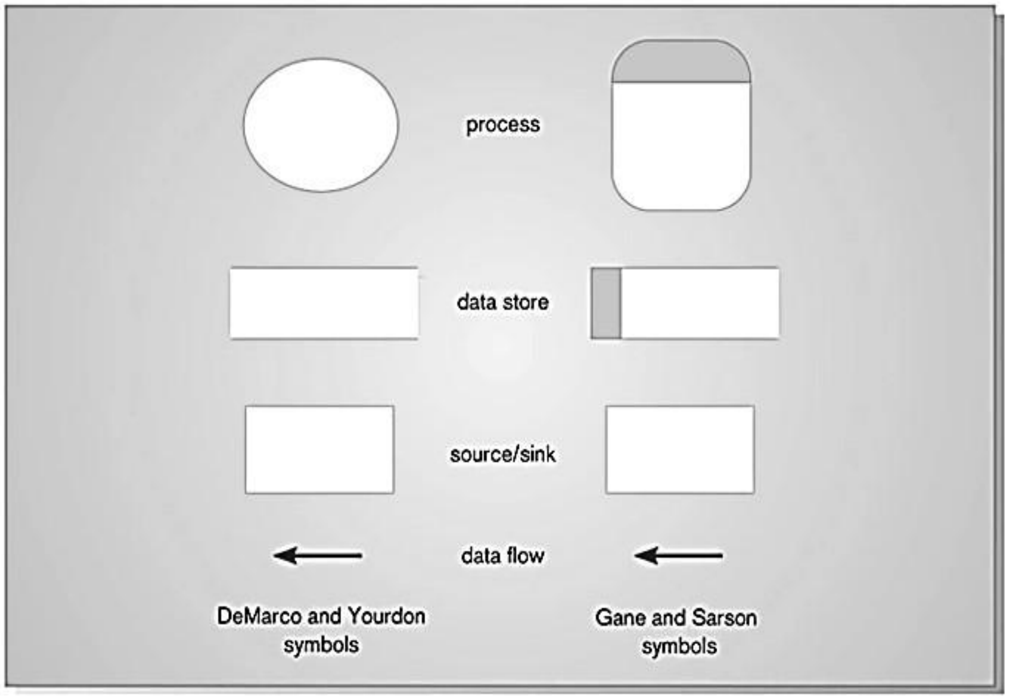
- ✓ DFD is likely to take many alteration before agreement with the user
- ✓ Physical consideration are usually left out
- ✓ It is difficult to understand because it ambiguous to the user who have little or no knowledge

Data Flow Diagramming Mechanics

Four symbols are used to represent DFDs (See Figure below). Two different standard sets can be used as proposed by:

- ✓ DeMarco and Yourdan
- ✓ Gane and Sarson

Figure Comparison of DeMarco and Yourdon and Gane and Sarson DFD symbol sets



Data Flow

- ✓ Shows data that are in motion from one place to another in the system.
- ✓ Drawn as an arrow
- ✓ Select a meaningful name to represent the data

Process

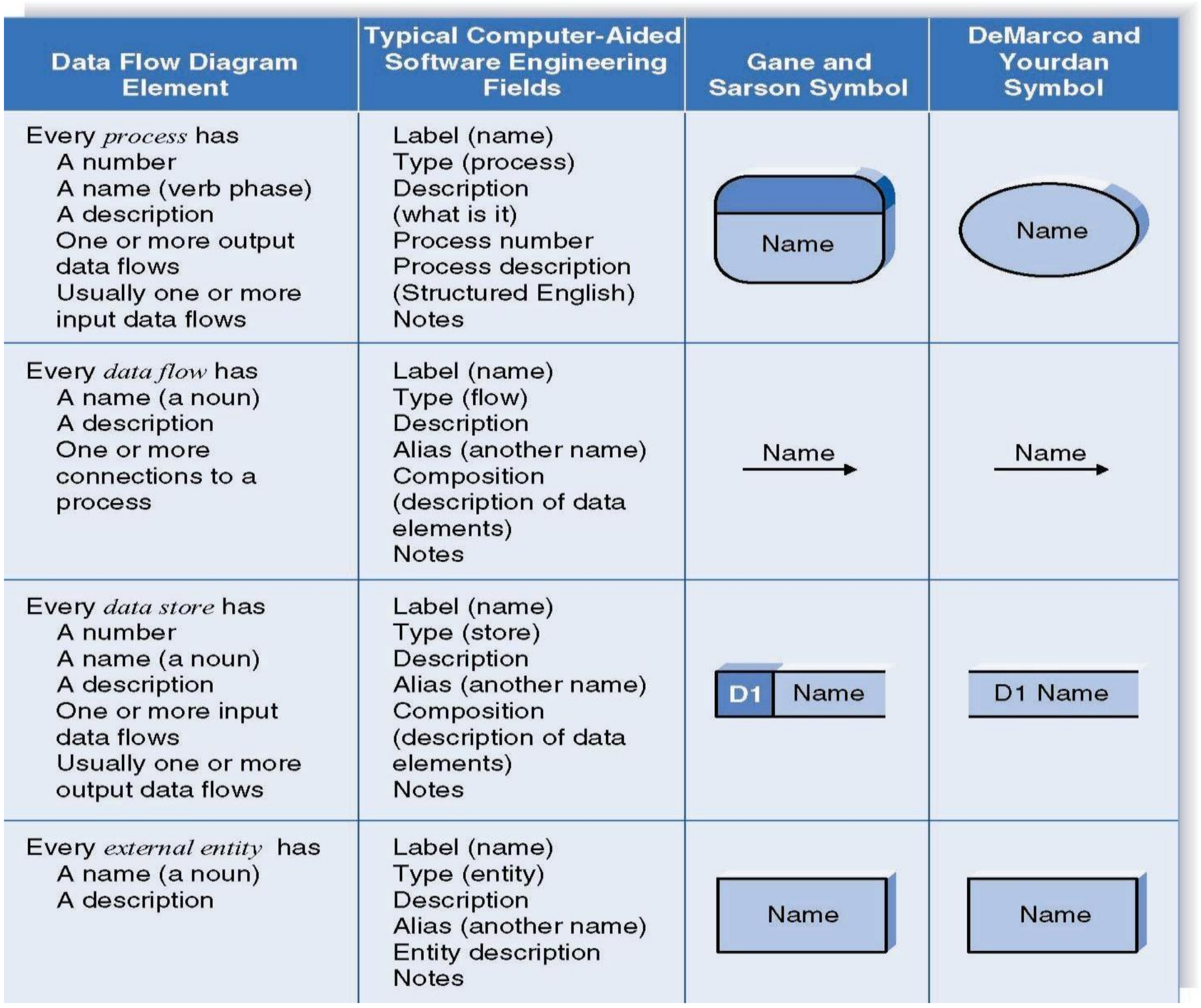
- ✓ Represents work or action performed on data so that they are transformed, stored or distributed (inside the system)
- ✓ Number of process as well as name are recorded
- ✓ Drawn as a rectangle with rounded corners

Data Store

- ✓ Represents data at rest (inside the system)
- ✓ May represent data in
  - File folder
  - Computer-based file
  - Notebook
- ✓ The name of the store as well as the number are recorded in between lines
- ✓ Drawn as a rectangle with the right vertical line missing
- ✓ Label includes name of the store as well as the number

Source/Sink

- ✓ Shows the origin and/or destination of the data (outside of system)
- ✓ Sometimes referred to as an external entity
- ✓ Drawn as a square symbol
- ✓ Name states what the external agent is
- ✓ Because they are external, many characteristics are not of interest to us
- ✓ A person, organization, or system that is external to the system but interacts with it.

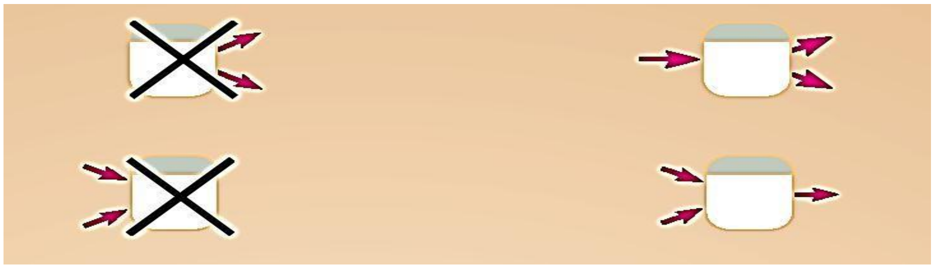


Data Flow Diagram Element	Typical Computer-Aided Software Engineering Fields	Gane and Sarson Symbol	DeMarco and Yourdan Symbol
Every *process* has A number A name (verb phase) A description One or more output data flows Usually one or more input data flows	Label (name) Type (process) Description (what is it) Process number Process description (Structured English) Notes	Name	Name
Every *data flow* has A name (a noun) A description One or more connections to a process	Label (name) Type (flow) Description Alias (another name) Composition (description of data elements) Notes	Name →	Name →
Every *data store* has A number A name (a noun) A description One or more input data flows Usually one or more output data flows	Label (name) Type (store) Description Alias (another name) Composition (description of data elements) Notes	D1 Name	D1 Name
Every *external entity* has A name (a noun) A description	Label (name) Type (entity) Description Alias (another name) Entity description Notes	Name	Name

DFD Diagramming Rules:

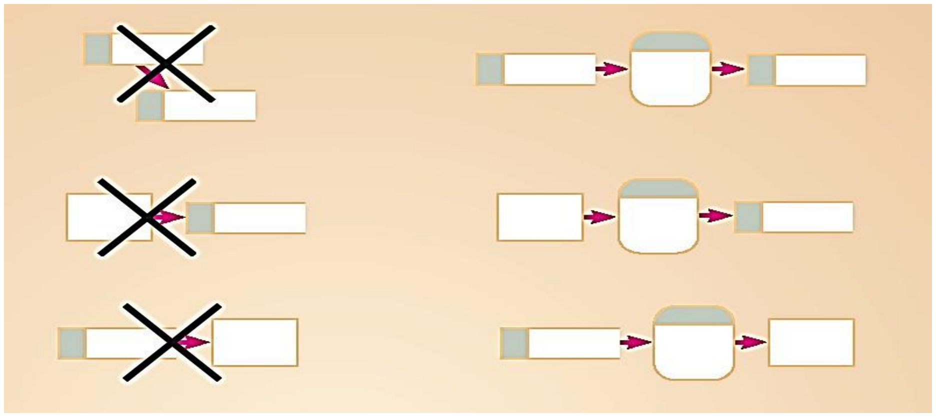
Rules for Process:

- ✓ No process can have only outputs or only inputs…processes must have both outputs and inputs.
- ✓ Process labels should be verb phrases.



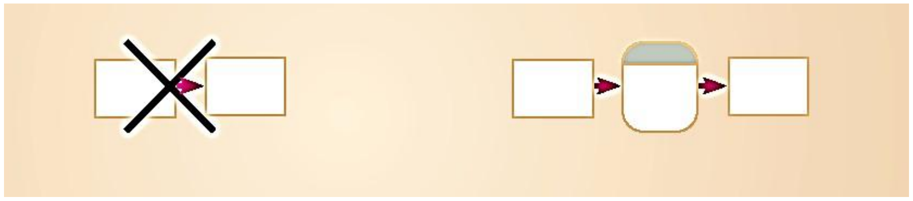
Rules for Data Store:

- ✓ All flows to or from a data store must move through a process.
- ✓ Data cannot be moved from one store to another
- ✓ Data cannot move from an outside source to a data store
- ✓ Data cannot move directly from a data store to a data sink
- ✓ Data store labels should be noun phrases.



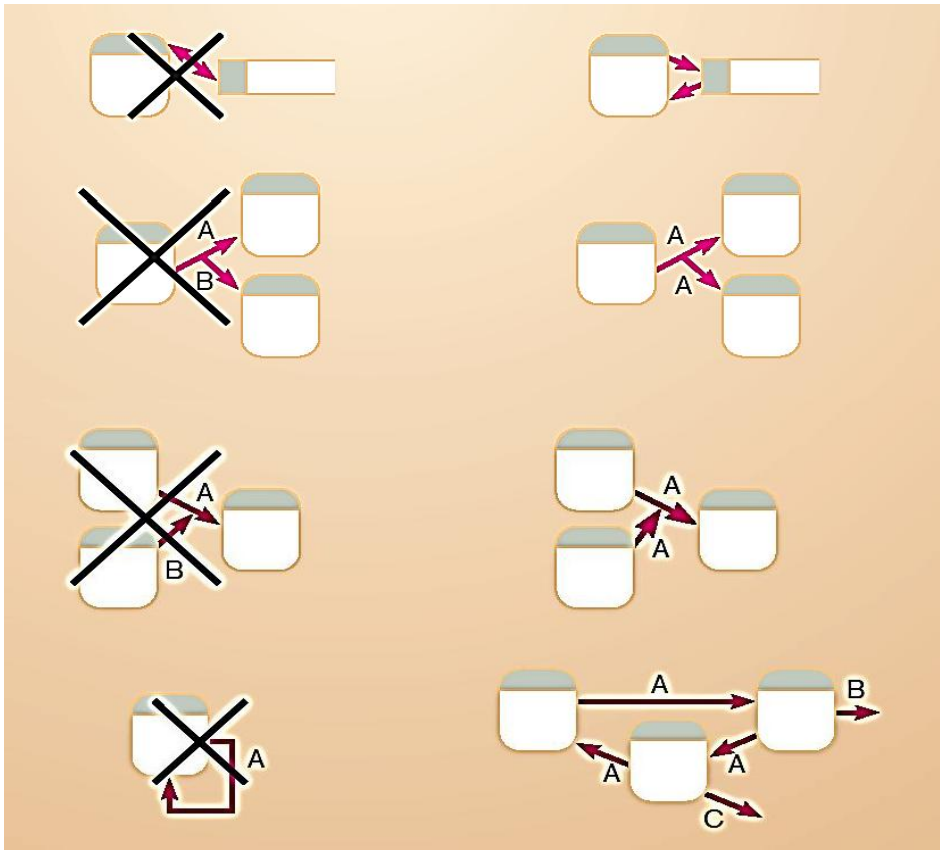
Rules for External Entities:

- ✓ 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.



Rules for Data Flow:

- ✓ A data flow has only one direction of flow between symbols
- ✓ Bidirectional flow between process and data store is represented by two separate arrows.
- ✓ A join means that exactly the same data come from any two or more different processes, data stores or sources/sinks to a common location.
- ✓ A fork means that exactly the same data go from a common location to two or more processes, data stores, or sources/sinks
- ✓ A data flow cannot go directly back to the same process it leaves
- ✓ A data flow to a data store means update
- ✓ A data flow from a data store means retrieve or use
- ✓ A data flow has a noun phrase label
- ✓ Forked data flow must refer to exact same data item (not different data items) from a common location to multiple destinations.
- ✓ 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.
- ✓ 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.



Functional Decomposition:

It is an iterative process of breaking a system description down into finer and finer detail, which creates a set of charts in which one process on a given chart is explained in greater detail on another chart.

Functional decomposition is an act of going from one single system to many component processes. It is a repetitive procedure. Decomposition goes on until you have reached the point where no sub process can logically be broken down any further. The lowest level of DFDs is called a primitive DFD.

DFD Levels:

- ✓ Context Diagram
  - 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

Context Diagram:

Context diagram shows the system boundaries, external entities that interact with the system, and major information flows between entities and the system. It is the first DFD in every business process. It shows the context into which the business process fits. It also shows the overall business process as just one process (process 0). It shows all the external entities that receive information from or contribute information to the system.

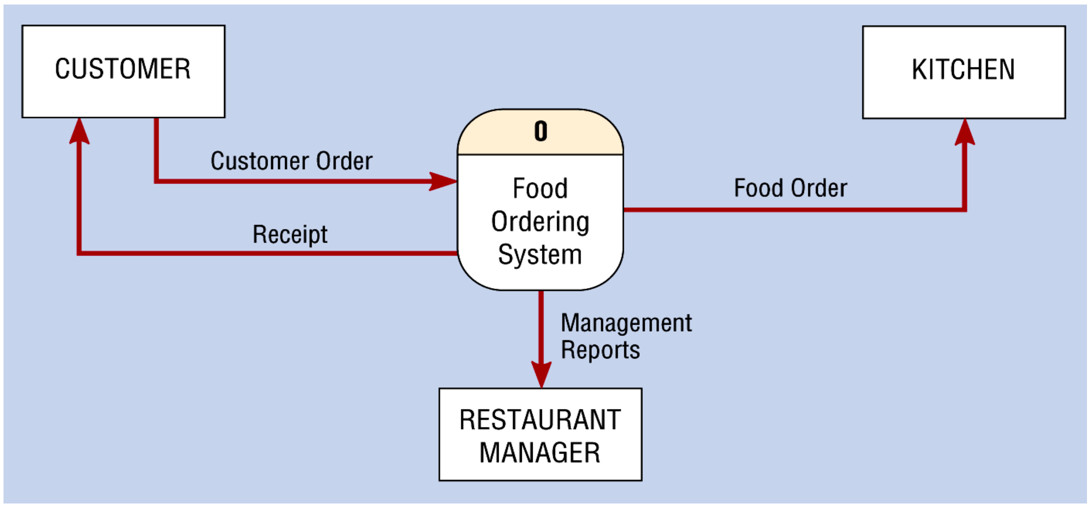


Figure: Context Diagram of Food Ordering System

Note: Only one process symbol and no data stores are shown in context diagram.

Level-0 DFD:

Level-0 DFD shows the system's major processes, data flows, and data stores at a high level of abstraction. It shows all the major processes that comprise the overall system – the internal components of process 0. It also shows how the major processes are interrelated by data flows. It shows external entities and the major processes with which they interact. It adds data stores.

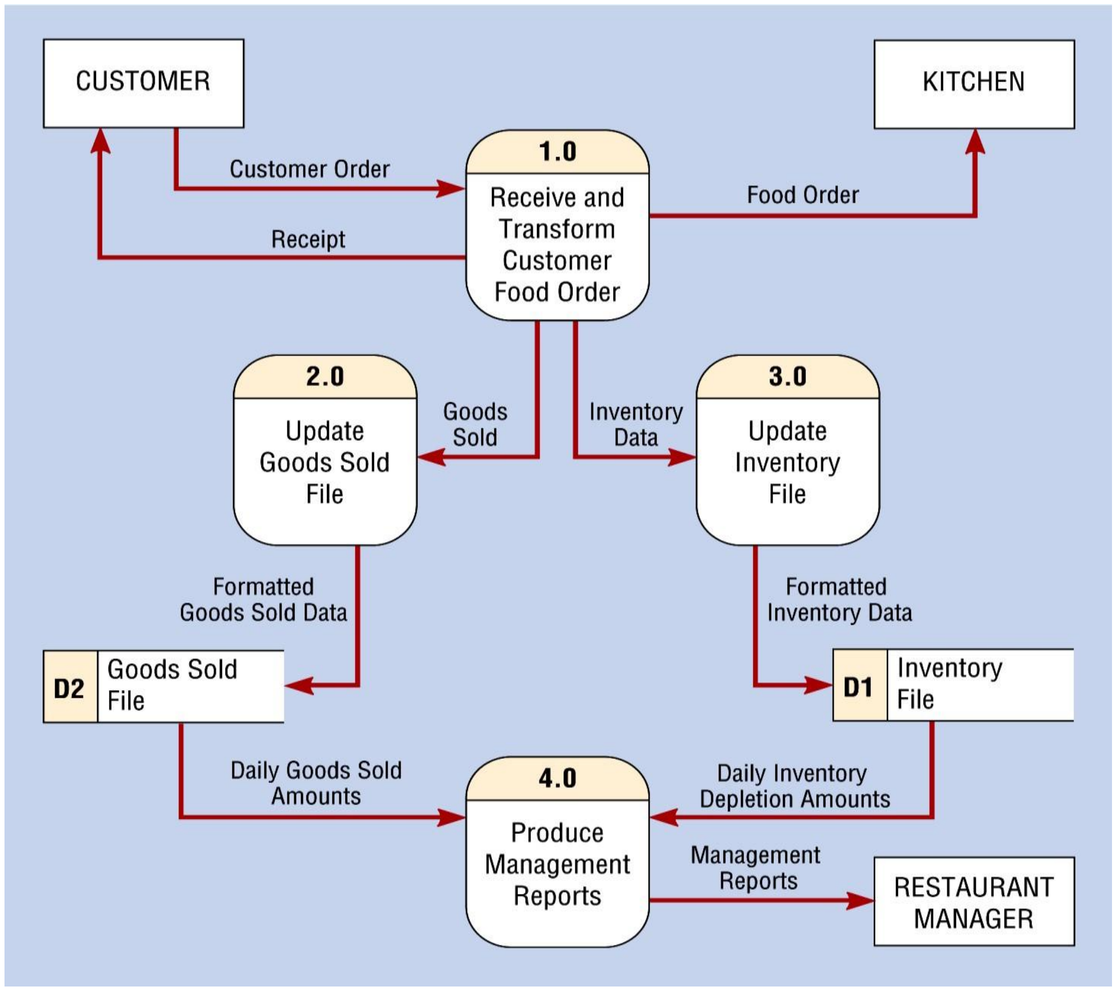


Figure: Level-0 DFD of Food Ordering System

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

Level-1 DFD:

Level-1 DFD shows the sub-processes of one of the processes in the Level-0 DFD. Generally, one level 1 diagram is created for every major process on the level 0 diagram. It shows all the internal processes that comprise a single process on the level 0 diagram. It also shows how information moves from and to each of these processes. If a parent process is decomposed into, for example, three child processes, these three child processes wholly and completely make up the parent process.

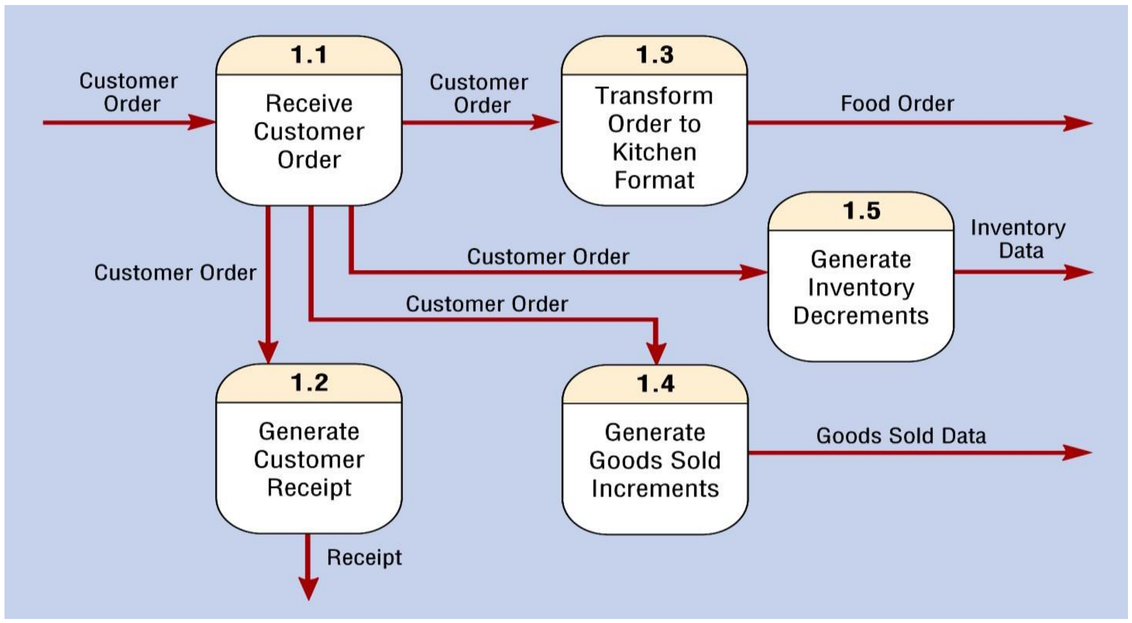


Figure: Level-1 Diagram Showing Decomposition of Process 1.0 from the Level-0 Diagram

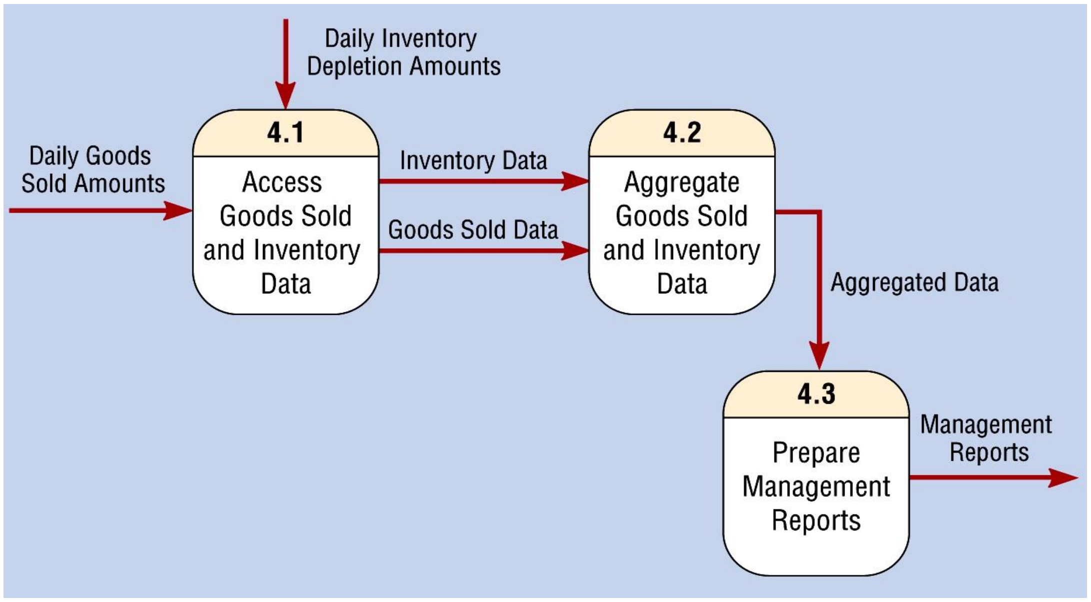

Daily Inventory Depletion Amounts
Daily Goods Sold Amounts
4.1
Access Goods Sold and Inventory Data
Inventory Data
Goods Sold Data
4.2
Aggregate Goods Sold and Inventory Data
Aggregated Data
4.3
Prepare Management Reports
Management Reports

Figure: Level-1 Diagram Showing the Decomposition of Process 4.0 from the Level-0 Diagram

Note: Processes are labeled 1.1, 1.2, 4.1, 4.2, etc. These can be further decomposed in more primitive (lower-level) DFDs if necessary. Sources and sinks are optional on level-1 diagrams.

Level-n DFD:

Level-n DFD shows the sub-processes of one of the processes in the Level n-1 DFD. It shows all processes that comprise a single process on the level 1 diagram. It also shows how information moves from and to each of these processes. Level 2 diagrams may not be needed for all level 1 processes. Correctly numbering each process helps the user understand where the process fits into the overall system.

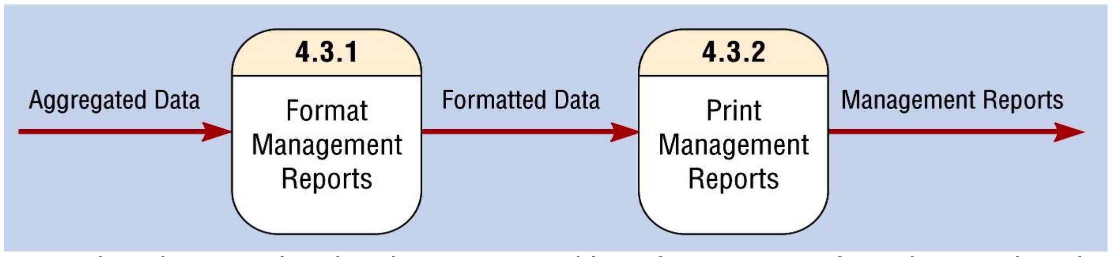
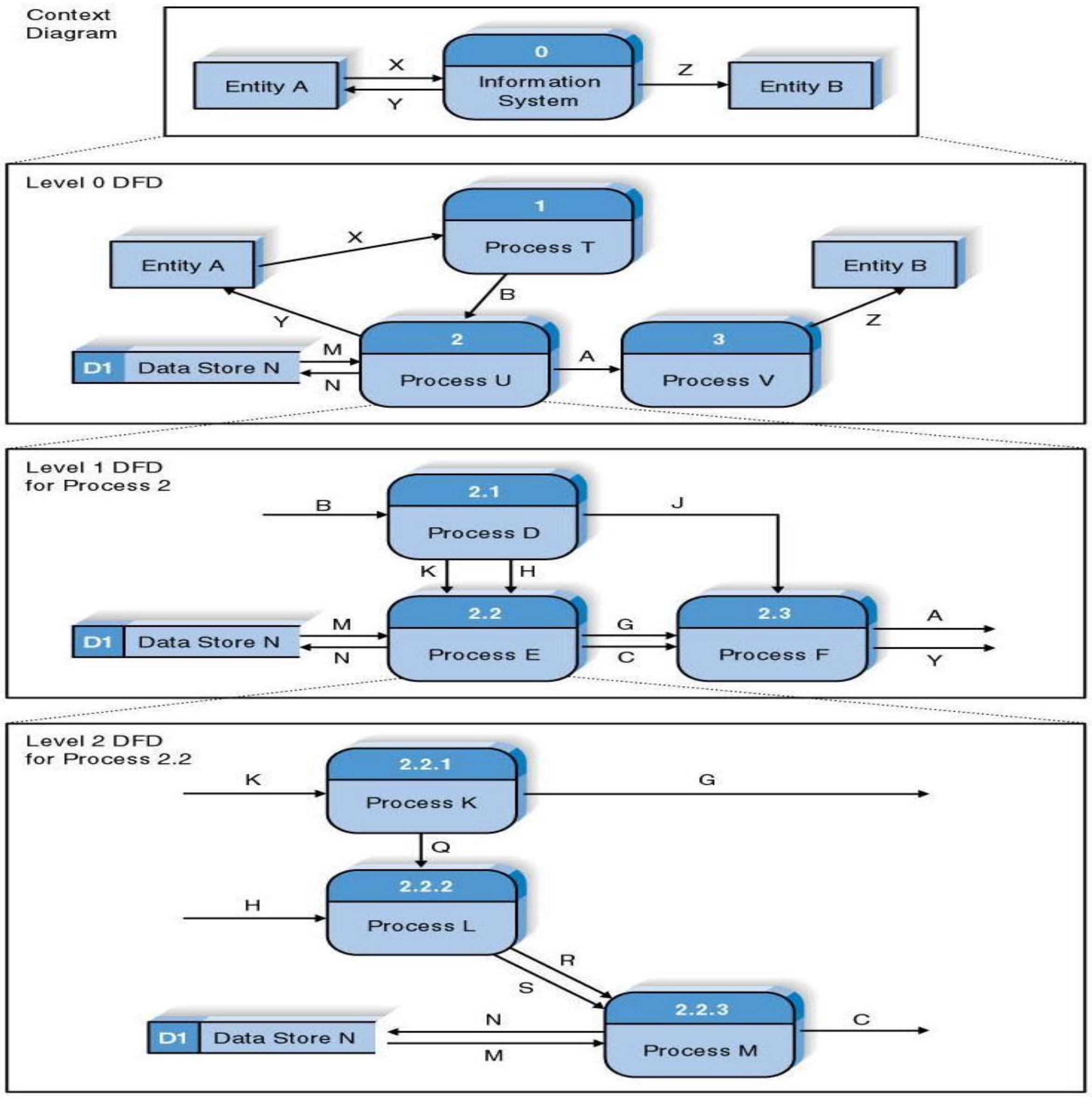


Figure: Level-2 Diagram Showing the Decomposition of Process 4.3 from the Level-1 Diagram for Process 4.0

Note: 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. Sources and sinks are optional on level-n diagrams.

Relationship among Levels of DFDs:



DFD Balancing:

When decomposing a DFD, you must conserve inputs to and outputs from a process at the next level of decomposition. This is called balancing.

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

Figure An unbalanced set of data flow diagrams - Context diagram

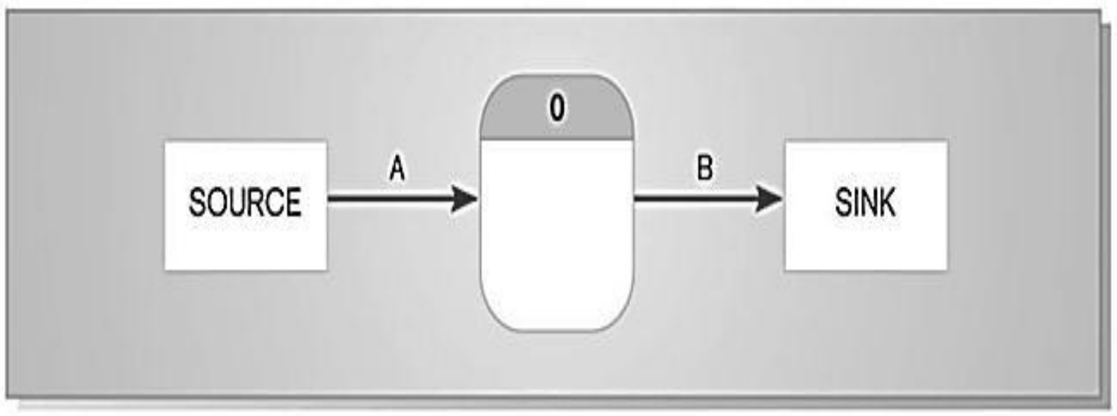
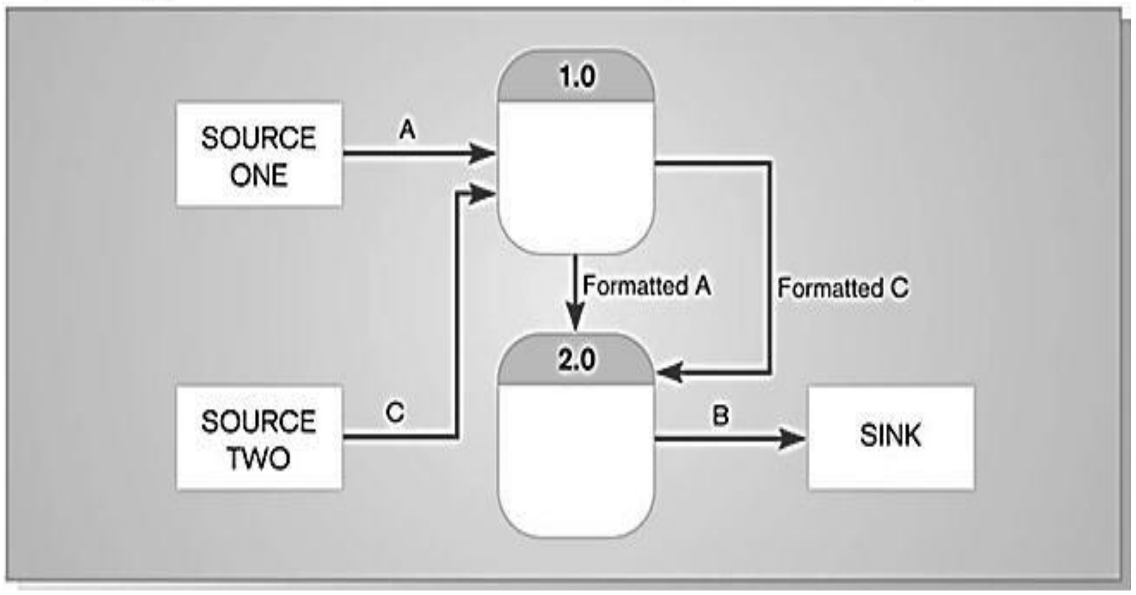


Figure An unbalanced set of data flow diagrams - Level-0 diagram



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

Figure Context diagram of food ordering system

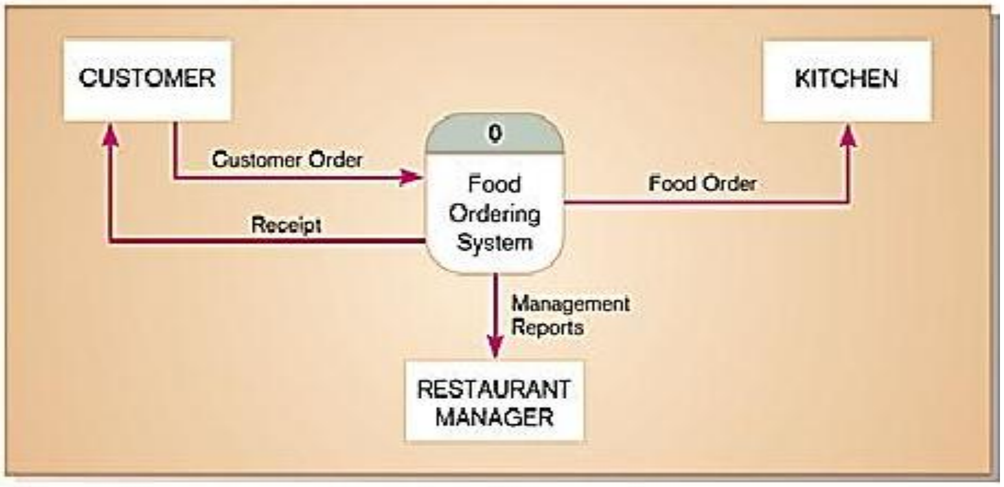
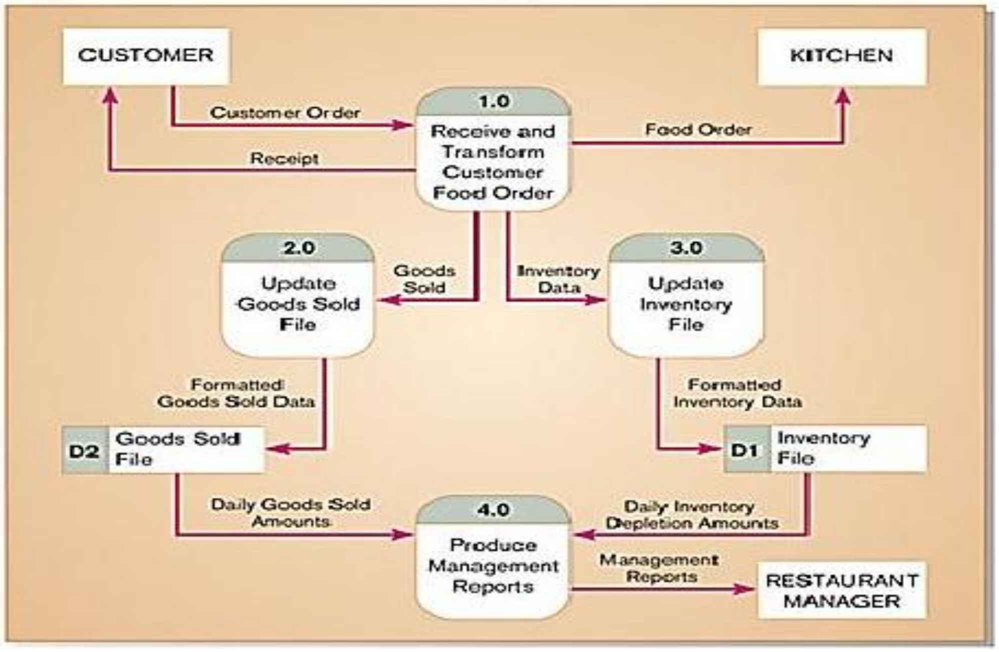


Figure Level-0 DFD of food ordering system



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.

When to stop decomposing DFDs?
Ideally, a DFD has at least three processes and no more than seven to nine.

How broad should the DFD be? (How many processes should be on a level?)
7 ± 2 is a reasonable heuristic.

How deep should be the DFD be? (How many levels should a DFD have?)
If the process has only one input or one output, probably cannot partition further.

Types of Data Flow Diagram:

Logical Data Flow Diagram:

Logical data flow model describe processes without suggesting how they are conducted. They are often more stable because they are based on business events and not on a particular technology or method of implementation. They have a business emphasis and help the analyst to understand the business being studied, to grasp why procedures are performed, and to determine the expected result of performing a task.

The data flow diagrams which represent the model of the proposed system are known as logical DFD. Logical data flow diagram represents business functions or processes. It describes the system independently of how it is actually implemented, and focuses rather on how an activity is accomplished.

Advantages in using a logical diagram:

- ✓ Better communication with users.
- ✓ More stable system.
- ✓ Better understanding of the business by analyst.
- ✓ Flexibility and maintenance.
- ✓ Elimination of redundancies and easier creation of the physical model

Physical Data Flow Diagram:

Physical data flow diagram shows how the system will be constructed. It provides information that is needed to build the system. It also have intermediate data stores-often, a transaction file.

The data flow diagrams which represent the model of the current system (manual or computerized), are known as physical DFD. These diagrams are drawn, when the analyst studies the current working system in detail.

Physical data flow diagrams are implementation-dependent and show the actual devices, department, people, etc., involved in the current system.

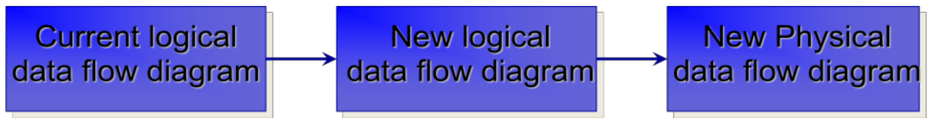
Advantages of physical data flow diagram:

- ✓ Clarifying which processes are manual and which are automated.
- ✓ Describing processes in more detail than do logical DFDs.
- ✓ Sequencing processes that have to be done in a particular order.
- ✓ Identifying temporary data stores.
- ✓ Specifying actual names of files and printouts.
- ✓ Adding controls to ensure the processes are done properly.

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 data flow diagram
New logical data flow diagram
New Physical data flow diagram

Current Logical:

Derive the logical data flow diagram for the current system by examining the physical data flow diagram and isolating unique business activities.

- ✓ Physical aspects of system are removed as much as possible.
- ✓ Current system is reduced to data and processes that transform them.

New Logical:

Create the logical data flow diagram for the new system by adding the input, output, and processes required in the new system to the logical data flow diagram for the current system.

- ✓ Includes additional functions
- ✓ Obsolete functions are removed
- ✓ Inefficient data flows are reorganized

New Physical:

Derive the physical data flow diagram by examining processes on the new logical diagram. Determine where the user interfaces should exist, the nature of the processes, and necessary data stores.

- ✓ Represents the physical implementation of the new system

Using Data Flow Diagramming in the Analysis Process

While drawing DFDs follow these guidelines

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.

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.

Primitive DFDs

- ✓ Lowest logical level of decomposition
- ✓ Decision has to be made when to stop decomposition

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
- ✓ When every data flow does not need to be split further to show that data are handled in various ways
- ✓ When you believe that you have shown each business form or transaction, online display and report as a single data flow
- ✓ When you believe that there is a separate process for each choice on all lowest-level menu options

Using DFDs as Analysis Tools

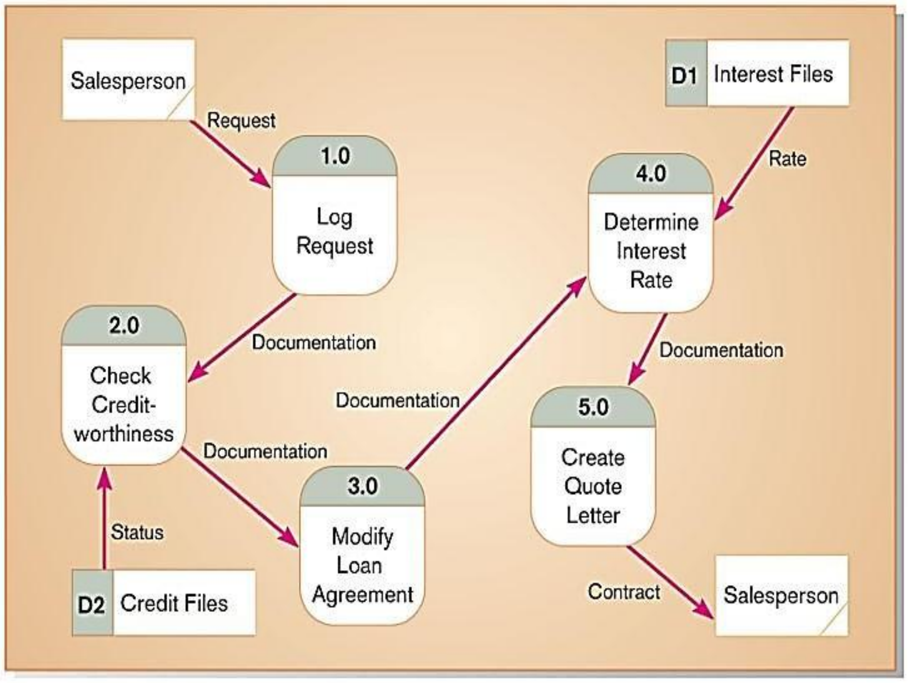
DFDs can be used in analysis for a process called gap analysis. It is the process of discovering discrepancies between two or more sets of data flow diagrams or discrepancies within a single DFD. Inefficiencies in a system can often be identified through DFDs.

Using DFDs in Business Process Reengineering:

Data flow diagrams also make a useful tool for modeling processes in business process reengineering (BPR). Let’s look on one example: IBM Credit Corporation.

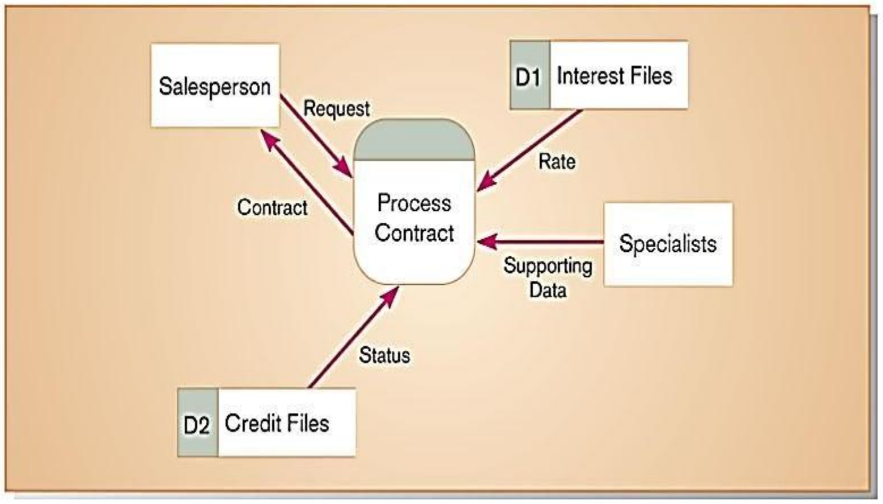
Credit approval process is required six days before Business Process Reengineering. After Business Reprocess Engineering, IBM was able to process 100 times the number of transactions in the same amount of time

Figure 7-20 IBM Credit Corporation's primary work process before BPR



Source: Copyright © 1993 Harper Business, an imprint of HarperCollins Publishers. Adapted with permission.

Figure 7-21 IBM Credit Corporation's primary work process after BPR



Source: Copyright © 1993 Harper Business, an imprint of HarperCollins Publishers. Adapted with permission.

References:

- ✓ Hoffer, J.A., George, J.F. and Valacich J.S., "Modern Systems Analysis and Design", 3rd Edition, Pearson Education, 2003.
- ✓ K.E.Kendall and J.E.Kendall, "Systems Analysis and Design", 5th Edition, Pearson Eduation, 2003.

- ✓ V.Rajaraman, “Analysis and Design of Information Systems”, 2nd Edition, Prentice Hall of India, New Delhi, 2002
- ✓ E.Yourdon “Modern Structured Analysis”, Prentice Hall of India, 1996.
- ✓ Elias M Awad, “Systems Analysis and Design”, Galgotia.
- ✓ Igor Hawryszkiewycz, “Systems Analysis and Design”, PHI.
- ✓ R.Schultheis and Mary Summer, “Management Information Systems”, Tata McGraw Hill, 1999.

Assignments:

(1) Develop a top-level DFD (Level-1 and Level-2) for Student Library System.
(2) List out some of the differences between Flowchart and DFDs.
(3) What is a good data flow diagram?
(4) Explain leveling convention for DFDs.
(5) How many levels of DFDs can be reached and why?

Gentle Advice:

Please go through your text books and reference books for detail study!!! Thank you all.