BACHELOR OF TECHNOLOGY YEAR 3

OBJECT-ORIENTED ANALYSIS AND DESIGN

WEEK 5

DATA AND PROCESS MODELING

During the requirements modeling process you used fact-finding techniques to investigate the current system and identify user requirements. Now we will use that information to develop a logical model of the proposed system and document the system requirements.

Logical Model

A *logical model* shows *what* the system must do, regardless of how it will be implemented physically.

Physical model

A *physical model* that describes *how* the system will be constructed

Data and process modeling involves three main tools:

- Data flow diagrams,
- Data dictionary,
- Process descriptions.

OVERVIEW OF DATA AND PROCESS MODELING TOOLS

Systems analysts use many graphical techniques to describe an information system. One popular method is to draw a set of data flow diagrams.

A *data flow diagram (DFD)* uses various symbols to show how the system transforms *Input* data into useful *information (Output)*. Other graphical tools include object models and entity-relationship diagrams, which we will look at later

DATA FLOW DIAGRAMS

A *data flow diagram (DFD)* shows how data moves through an information system but does not show program *logic* or *processing* steps. A set of DFDs provides a logical model that shows *what* the system does, not *how* it does it. This distinction is important because focusing on implementation issues at this point would restrict your search for the most effective system design.

DFD SYMBOLS

DFDs use four basic symbols that represent processes, data flows, data stores, and entities.

a) Process Symbol

A *process* receives *input* data and produces *output* that has a different content, form, or both.

Example

The process for calculating pay uses two inputs (pay rate and hours worked) to produce one output (*total pay*). Processes can be very simple or quite complex. In a typical company, processes might include calculating sales trends, filing online insurance claims, ordering inventory from a supplier's system, or verifying e-mail addresses for Web customers.

Processes contain the **business logic**, also called **business rules** that transform the data and produce the required results.

The symbol for a process is a rectangle with rounded corners. The name of the process appears inside the rectangle. The process name *identifies* a specific function and consists of a *verb* (and an adjective, if necessary) followed by a singular noun.

Examples of process names are *APPLY RENT PAYMENT*, *CALCULATE COMMISSION*, *ASSIGN FINAL GRADE*, *VERIFY ORDER*, and *FILL ORDER.*

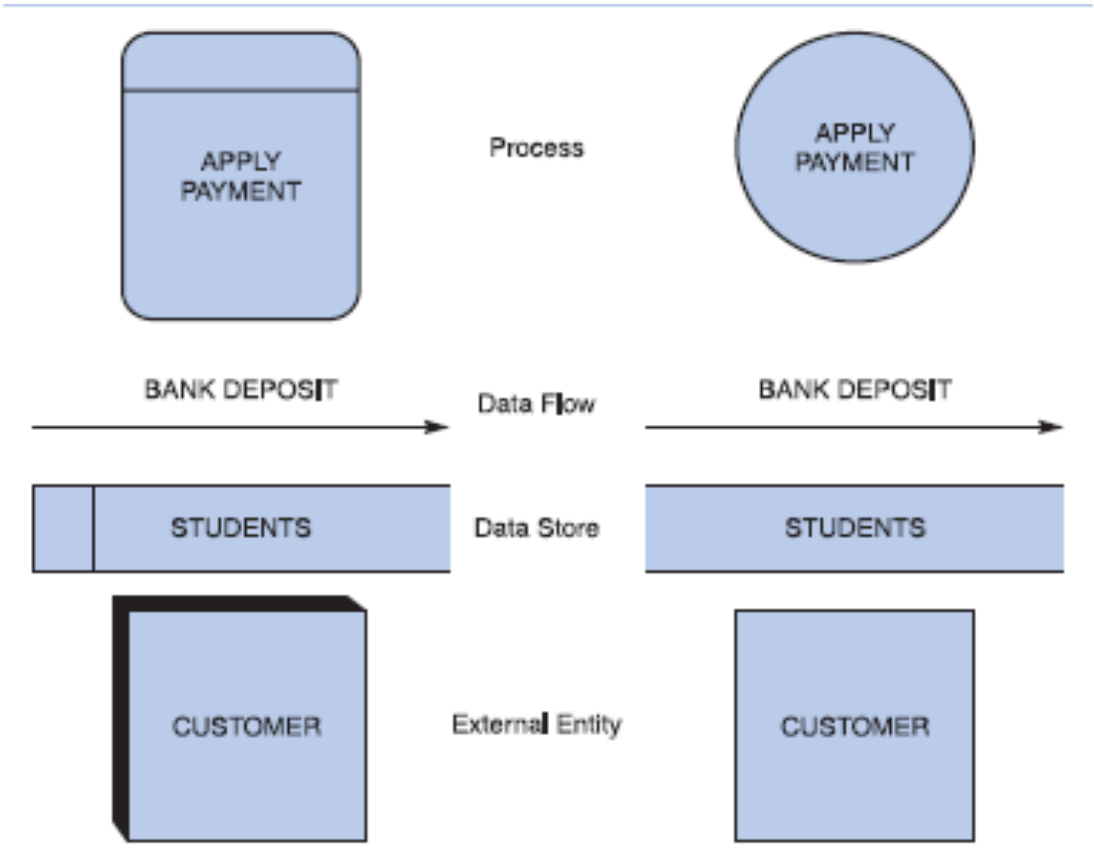
NOTE:-

Processing details are not shown in a *DFD*. That is, you might have a process named DEPOSIT PAYMENT. The process symbol does not reveal the business logic for the DEPOSIT PAYMENT process.

In DFDs, a process symbol can be referred to as a **black box**, because the inputs, outputs, and general functions of the process are known, but the underlying details and logic of the process are hidden.

By showing processes as black boxes, an analyst can create DFDs that show how the system functions, but avoid unnecessary detail.

When the analyst wishes to show additional levels of detail, he or she can zoom in on a process symbol and create a more in-depth DFD that shows the process's internal workings — which might reveal even more processes, data flows, and data stores. In this manner, the information system can be modeled as a series of increasingly detailed diagrams.



b) Data Flow Symbol

A *data flow* is a path for data to move from *one part* of the information system *to another*. A data flow in a DFD represents one or more data items. For example, a data flow could consist of a single data item (such as a student ID number) or it could include a set of data (such as a class roster with student ID numbers, names, and registration dates for a specific class).

The symbol for a data flow is a *line* with a *single* or *double* arrowhead. The data flow name appears above, below, or alongside the line. A data flow name consists of a *singular* noun and an adjective if needed.

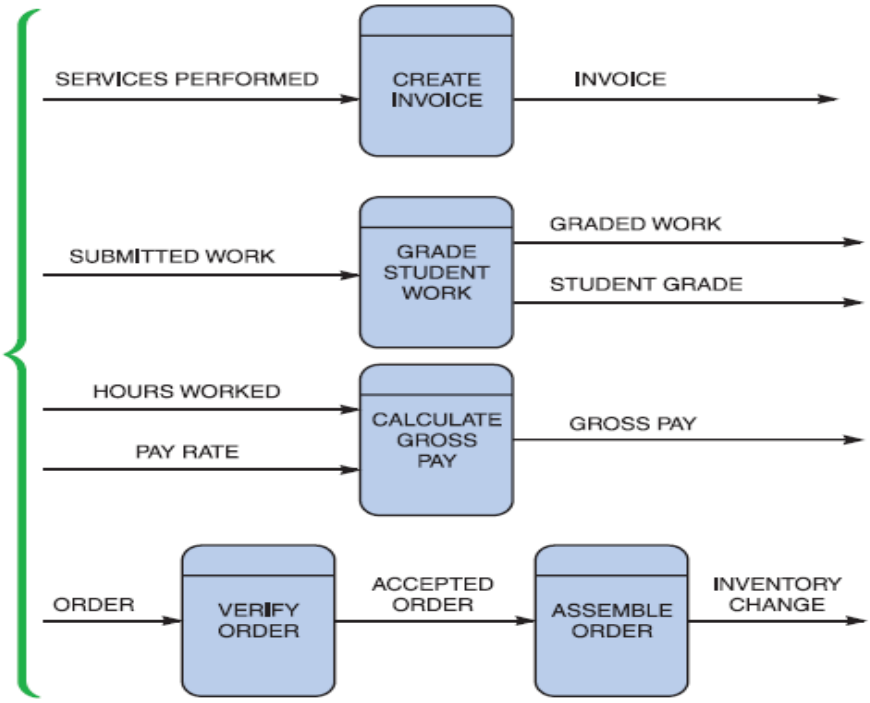
Examples of data flow names are *DEPOSIT*, *INVOICE PAYMENT*, *STUDENT GRADE*, *ORDER*, and *COMMISSION*.

The figure below shows correct examples of data flow and process symbol connections. Because a process changes the data's content or form, at least one data flow must enter and one data flow must exit each process symbol, as they do in the *CREATE INVOICE* process.

A process symbol can have more than one outgoing data flow, as shown in the G*RADE STUDENT WORK* process, or more than one incoming data flow, as shown in the *CALCULATE GROSS PAY* process.

A process also can connect to any other symbol, including another process symbol, as shown by the connection between VERIFY ORDER and ASSEMBLE ORDER as shown below.

A data flow, therefore, *must* have a process symbol on at least one end.



Combinations of data flow and process symbols

c) Data Store Symbol

A *data store* is used in a DFD to represent data that the *system stores* because one or more processes need to use the data at a later time. For instance, instructors need to store *student scores* on tests and assignments during the semester so they can assign *final grades* at the end of the term.

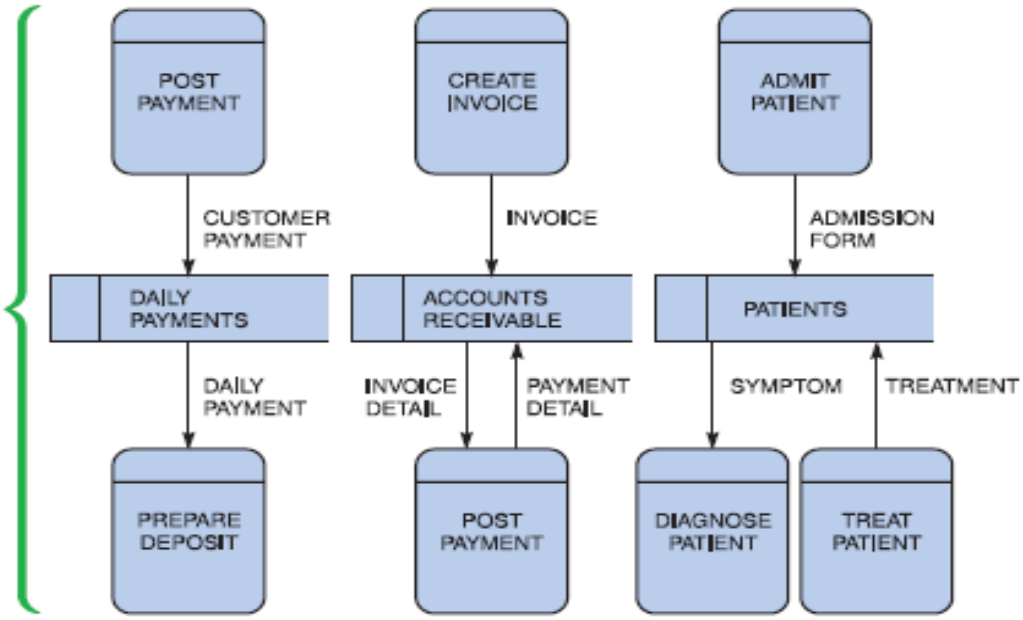
A DFD does not show the detailed contents of a data store ; the *specific structure* and *data elements* are defined in the *data dictionary*.

The physical characteristics of a data store are unimportant because you are concerned only with a *logical model.*

In a DFD, the symbol for a data store is a *flat rectangle* that is *open* on the *right side* and closed on the *left side*.
The name of the data store appears between the lines and *identifies* the data it contains. A data store name is a plural name consisting of a noun and adjectives if needed.

Example

Examples of data store names are STUDENTS, ACCOUNTS RECEIVABLE, PRODUCTS, DAILY PAYMENTS, PURCHASE ORDERS, OUTSTANDING CHECKS, INSURANCE POLICIES, and EMPLOYEES. Exceptions to the plural name rule are collective nouns that represent multiple occurrences of objects. For example, GRADEBOOK represents a group of students and their scores.



A data store must be connected to a process with a data flow. The figure above illustrates typical examples of data stores. In each case, the data store has at least one *incoming* and one *outgoing* data flow and is connected to a process symbol with a data flow.

Entity Symbol

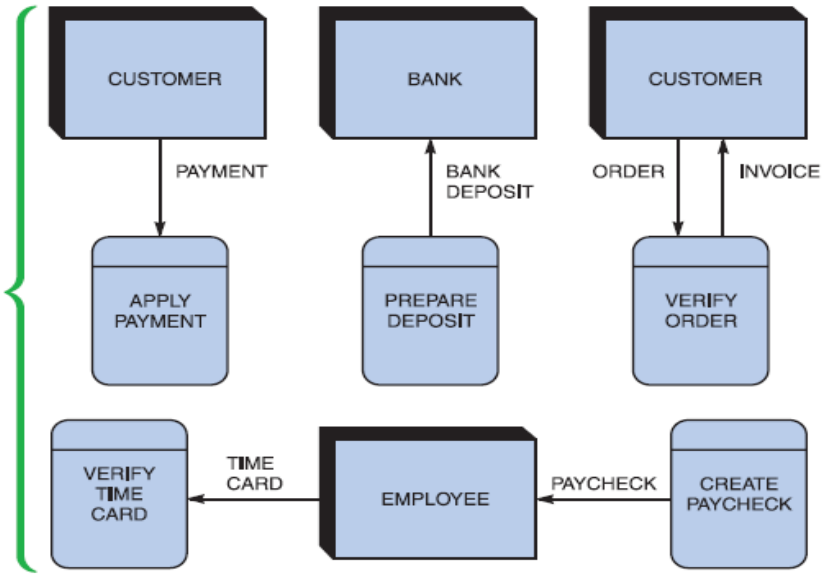
The symbol for an *entity* is a rectangle, which may be shaded to make it look three-dimensional. The name of the entity appears inside the symbol.

A DFD shows only *external entities* that *provide* data to the system or *receive output* from the system. A DFD shows the boundaries of the system and how the system interfaces with the outside world.

For example, a customer entity *submits an order* to an order processing system. Other examples of entities include a *patient who supplies data* to a medical records system, a *homeowner who receives a bill* from a city property tax system, or an accounts payable system that receives data from the *company's purchasing system*.

DFD entities also are called *terminators*, because they are data origins or final destinations. Systems analysts call an entity that supplies data to the system a *source*, and an entity that receives data from the system a *sink*.

An entity name is the *singular* form of a department, outside organization, other information systems, or person. An external entity can be a source or a sink or both, but each entity must be connected to a process by a data flow as shown below



CREATING A SET OF DFDS

During requirements modeling, you used *interviews*, *questionnaires*, and other techniques to *gather facts* about the system, and you learned how the various people, departments, data, and processes fit together to support business operations.

Now you are ready to create a *graphical model* of the information system based on your fact-finding results.

GUIDELINES FOR DRAWING DFDS

When drawing DFDs, you should follow guidelines:-

- Draw the context diagram so it fits on one page.
- Use the name of the information system as the process name in the context diagram.
- Use unique names within each set of symbols.
- Do not cross lines. One way to achieve that goal is to restrict the number of symbols in any DFD.

How to Avoid Cross Lines

  - On lower-level diagrams with multiple processes, you should not have more than nine process symbols. Including more than nine symbols usually is a signal that your diagram is too complex and that you should reconsider your analysis.
  - Another way to avoid crossing lines is to duplicate an entity or data store.
  - When duplicating a symbol on a diagram, make sure to document the duplication to avoid possible confusion.
- Provide a unique name and reference number for each process.
- Obtain as much user input and feedback as possible. Your main objective is to ensure that the model is accurate, easy to understand, and meets the needs of its users.

CONTEXT DIAGRAM

Because it is the highest-level DFD, the context diagram contains *process 0*, which represents the entire information system, but does not show the internal workings.

To describe the next level of detail inside process 0, you must create a *DFD named diagram 0,* which will reveal additional processes that must be named and numbered. As you continue to create lower-level DFDs, you assign unique names and *reference numbers* to all processes, until you complete the *logical model*.

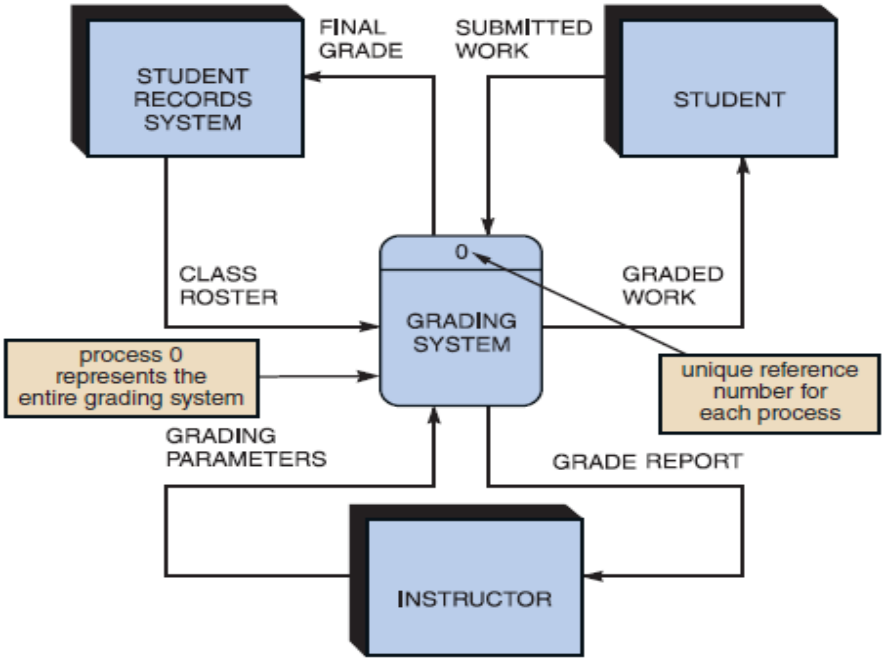
Example One:

The context diagram for a grading system is shown in the figure below and the *GRADING SYSTEM* process is at the center of the diagram.

The three entities (*STUDENT RECORDS SYSTEM, STUDENT,* and *INSTRUCTOR*) are placed around the central process. Interaction among the central process and the entities involves six different data flows.

The *STUDENT RECORDS SYSTEM* entity supplies data through the *CLASS ROSTER* data flow and receives data through the *FINAL GRADE* data flow.

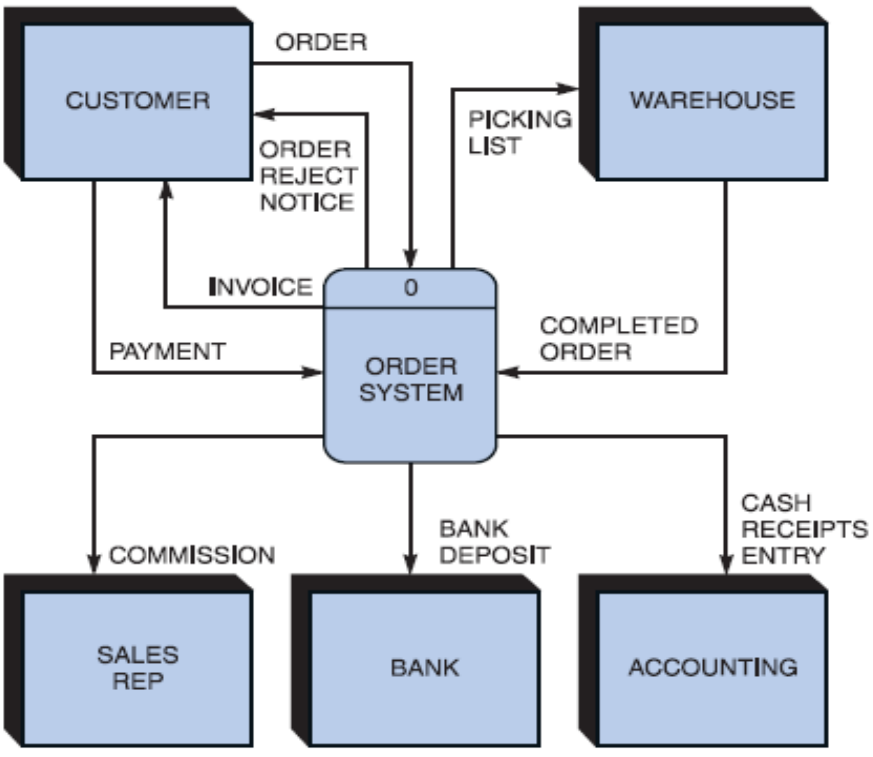
The *STUDENT* entity supplies data through the *SUBMITTED WORK* data flow and receives data through the *GRADED WORK* data flow. Finally, the *INSTRUCTOR* entity supplies data through the *GRADING PARAMETERS* data flow and receives data through the *GRADE REPORT* data flow.



Example Two

The context diagram for an order system is shown below. Note that the *ORDER SYSTEM* process is at the center of the diagram and five entities surround the process. Three of the entities, *SALES REP*, *BANK*, and *ACCOUNTING*, have single incoming data flows for COMMISSION, BANK DEPOSIT, and CASH RECEIPTS ENTRY, respectively.

The *WAREHOUSE* entity has one incoming data flow — *PICKING LIST* — that is, a report that shows the items ordered and their quantity, location, and sequence to pick from the warehouse. The WAREHOUSE entity has one outgoing data flow, *COMPLETED ORDER*. Finally, the *CUSTOMER* entity has two outgoing data flows, *ORDER* and *PAYMENT*, and two incoming data flows, *ORDER REJECT NOTICE* and *INVOICE*. The context diagram has two more entities and three more data flows. What makes one system more complex than another is the number of *components*, the number of levels, and the degree of interaction among its *processes*, *entities*, *data stores*, and *data flows*.

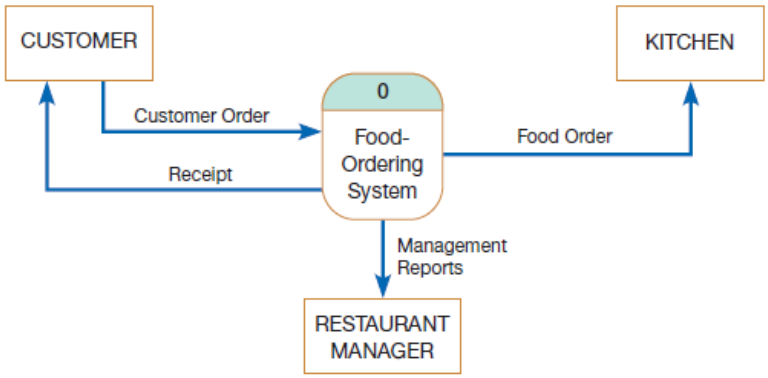


CREATING DFDs

The information system is depicted as a DFD in Figure 7-4. The highest-level view of this system, shown in the figure, is called a *context diagram*.

You will notice that this context diagram contains only one process, no data stores, four data flows, and three sources/sinks. The single process, labeled 0, represents the entire system; all context diagrams have only one process, labeled 0.

The sources/sinks represent the environmental boundaries of the system. Because the data stores of the system are conceptually inside one process, data stores do not appear on a context diagram.



DFD LEVEL 0

Decompose the *Food Ordering Systems* to subsystem and four subsystems are identified as shown below:-

Notice that the sources/sinks are the same in the context diagram and in this diagram: the customer, the kitchen, and the restaurant's manager.

This diagram is called a *level-0 diagram* because it represents the primary individual processes in the system at the highest possible level.
Each process has a number that ends in .0 (corresponding to the level number of the DFD).

Two of the data flows generated by the first process, *Receive* and *Transform Customer Food Order*, go to external entities, so we no longer have to worry about them. We are not concerned about what happens outside our system.

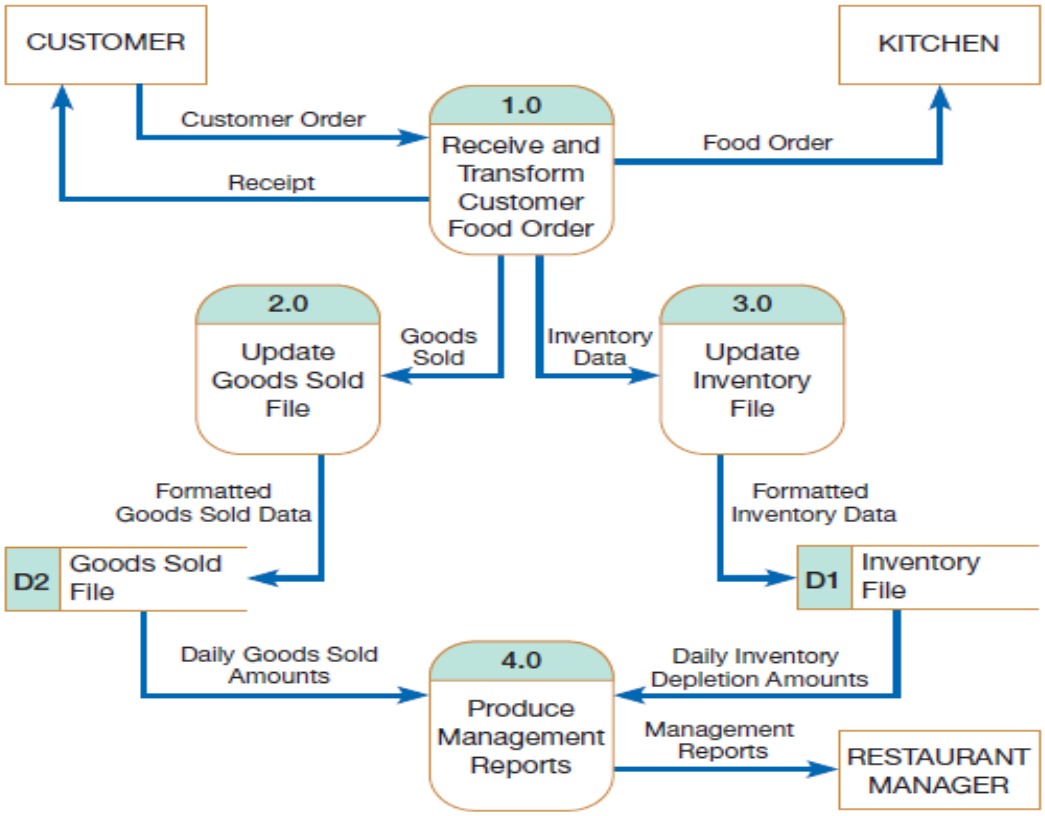
Let's trace the flow of the data represented in the other two data flows. First, the data labeled Goods Sold go to Process 2.0, *Update Goods Sold File*.

The output for this process is labeled *Formatted Goods Sold Data*. This output updates a data store labeled

Goods Sold File. If the customer order was for two cheeseburgers, one order of fries, and a large soft drink, each of these categories of goods sold in the data store would be incremented appropriately.

The Daily Goods Sold Amounts are then used as input to *Process 4.0*, Produce Management Reports.

Similarly, the remaining data flow generated by *Process 1.0*, Inventory Data, serves as *input* for *Process 3.0*, *Update Inventory File*.



This process updates the *Inventory File data store*, based on the inventory that would have been used to create the *customer order*.

The Daily Inventory Depletion Amounts are then used as input to Process 4.0. The data flow leaving Process 4.0, *Management Reports*, goes to the sink Restaurant Manager.

Notice that the sources/sinks are the same in the context diagram and in this diagram: the customer, the kitchen, and the restaurant's manager.

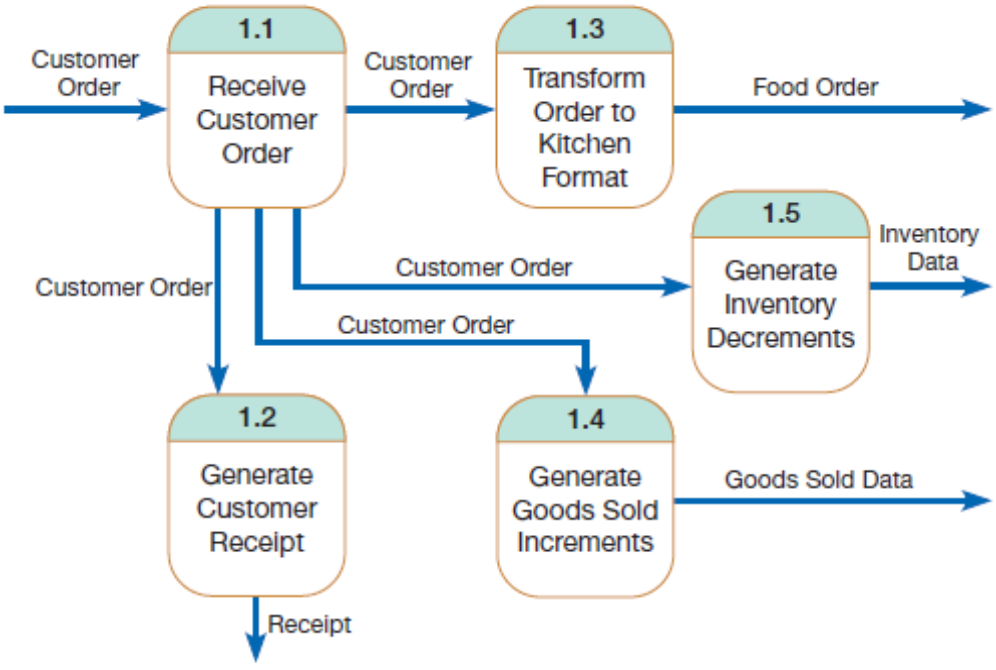
DFD LEVEL 1

Example 1

These major functions often correspond to the activities on the main system menu. We see that the system begins with an order from a customer, as was the case with the context diagram.

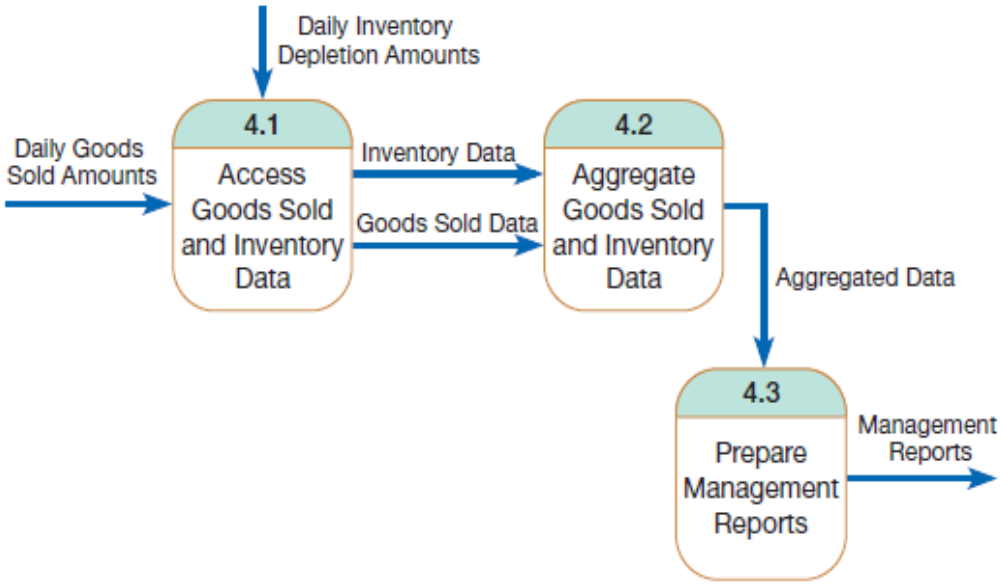
In the first process, labeled 1.0, we see that the customer order is processed. The result is four streams, or flows, of data:

- the food order is transmitted to the kitchen
- the customer order is transformed into a list of goods sold
- the customer order is transformed into inventory data
- The process generates a receipt for the customer.



Example 2

Level-1 diagram showing the decomposition of Process 4.0 from the level-0 diagram for the food-ordering system



DFD LEVEL 2

Again, notice how the sub-processes are labeled. Just as the labels for processes must follow *numbering rules* for clear communication, process names should also be *clear* yet *concise*.

Typically, process names begin with an action verb, such as Format, Print etc.