SHELLY CASHMAN SERIES®

Systems Analysis and Design

TENTH EDITION

ROSENBLATT

CHAPTER 5 Data and Process Modeling

Chapter 5 is the second of four chapters in the systems analysis phase of the SDLC. This chapter discusses data and process modeling techniques that analysts use to show how the system transforms data into useful information. The deliverable, or end product, of data and process modeling is a logical model that will support business operations and meet user needs.

INTRODUCTION

OBJECTIVES

When you finish this chapter, you will be able to:

- Describe data and process modeling concepts and tools, including data flow diagrams, a data dictionary, and process descriptions
- Describe the symbols used in data flow diagrams and explain the rules for their use
- Draw data flow diagrams in a sequence, from general to specific
- Explain how to level and balance a set of data flow diagrams
- Describe how a data dictionary is used and what it contains
- Use process description tools, including structured English, decision tables, and decision trees
- Describe the relationship between logical and physical models

During the requirements modeling process described in Chapter 4, you used fact-finding techniques to investigate the current system and identify user requirements. Now, in Chapters 5 and 6 you will use that information to develop a logical model of the proposed system and document the system requirements. A **logical model** shows *what* the system must do, regardless of how it will be implemented physically. Later, in the systems design phase, you build a **physical model** that describes *how* the system will be constructed. Data and process modeling involves three main tools: data flow diagrams, a data dictionary, and process descriptions.

If you have MIS CourseMate, you can view four Video Learning Sessions that explain how to work with DFD symbols and diagrams, how to create a context diagram, how to create a diagram 0 DFD, and how to use a decision table.

PREVIEW CASE: Mountain View College Bookstore

Background: Wendy Lee, manager of college services at Mountain View College, wants a new information system that will improve efficiency and customer service at the three college bookstores.

In this part of the case, Tina Allen (systems analyst) and David Conroe (student intern) are talking about data and process modeling tasks and concepts.

Participants:	Tina and David
Location:	Tina's office, Thursday afternoon, October 17, 2013
Project status:	Tina and David are ready to develop a requirements model that will describe and document the proposed system
Discussion topics:	Data flow diagrams, data dictionaries, and process description tools

Tina: Hi, David. Any questions about the fact-finding we did?

David: *Well, I found out that fact-finding is hard work.*

Tina: Yes, but it was worth it. Look at what we learned — now we understand how the current system operates, and we know what users expect in the new system. This information will help us build a requirements model that we can present to Wendy and her staff.

David: *What's the next step?*

Tina: We need to draw a set of data flow diagrams, or DFDs for short.

David: *Do we use a CASE tool to draw the DFDs?*

Tina: We can draw the initial versions by hand and use a CASE tool to prepare the final diagrams.

David: *What goes into a DFD?*

Tina: DFDs use four basic symbols that represent processes, data flows, data stores, and entities. You'll learn about these as we go along. I'll also show you how we use techniques called leveling and balancing to develop accurate, consistent DFDs.

David: *Apart from the diagrams, do we need to develop any other documentation?*

Tina: Yes, we need to create a data dictionary and process descriptions. The data dictionary is an overall storehouse of information about the system, and serves as a central clearinghouse for all documentation. We use process descriptions to explain the logical steps that each process performs. To create these descriptions, we use three tools: structured English statements, decision tables, and decision trees.

David: *Sounds like a lot to do. Where do we begin?*

Tina: Here's a task list to get us started:

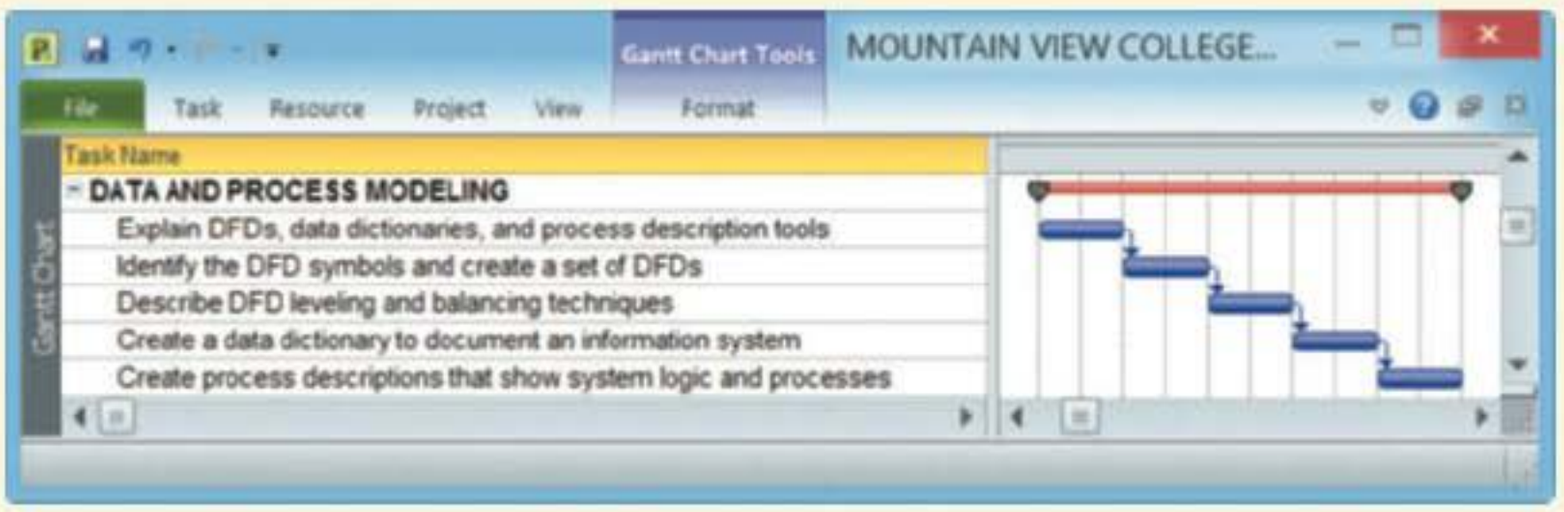


FIGURE 5-1 Typical data and process modeling task list.

Overview of Data and Process Modeling Tools

TOOLKIT TIME

The CASE Tools in Part B of the Systems Analyst's Toolkit can help you document business functions and processes, develop graphical models, and provide an overall framework for information system development. To learn more about these tools, turn to Part B of the four-part Toolkit that follows Chapter 12.

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. Other graphical tools include object models, which are explained in Chapter 6 (Object Modeling), and entity-relationship diagrams, which are described in Chapter 9 (Data Design).

Data Flow Diagrams

In Part A of the Systems Analyst's Toolkit, you learn how to use visual aids to help explain a concept, as shown in Figure 5-2. Similarly, during the systems analysis phase, you learn how to create a visual model of the information system using a set of 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. That distinction is important because focusing on implementation issues at this point would restrict your search for the most effective system design.

FIGURE 5-2 Systems analysts often use visual aids during presentations.

Courtesy of Luidia, Inc.

Video Learning Session
DFD Symbols and Diagrams

If you have an MIS CourseMate access code, you can launch interactive Video Learning Sessions to help you understand systems development concepts and practice your skills. You can watch the sessions on your computer or mobile device, and pause, rewind, or replay a video at any time. To log on to the MIS CourseMate site at **www.cengagebrain.com**, you must create a student account and then register this book.

This session is about DFD symbols and diagrams. You'll learn why DFDs are important modeling tools, how to use DFD symbols, and how you can use a CASE tool to create DFDs.

© craftvision/iStockphoto

DFD Symbols

DFDs use four basic symbols that represent processes, data flows, data stores, and entities. Several different versions of DFD symbols exist, but they all serve the same purpose. DFD examples in this textbook use the **Gane and Sarson** symbol set. Another popular symbol set is the **Yourdon** symbol set. Figure 5-3 shows examples of both versions. Symbols are referenced by using all capital letters for the symbol name.

PROCESS SYMBOL A process receives input data and produces output that has a different content, form, or both. For instance, 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.

Processing details are not shown in a DFD. For example, you might have a process named DEPOSIT PAYMENT. The process symbol does not reveal the business logic for the DEPOSIT PAYMENT process. To document the logic, you create a process description, which is explained later in this chapter.

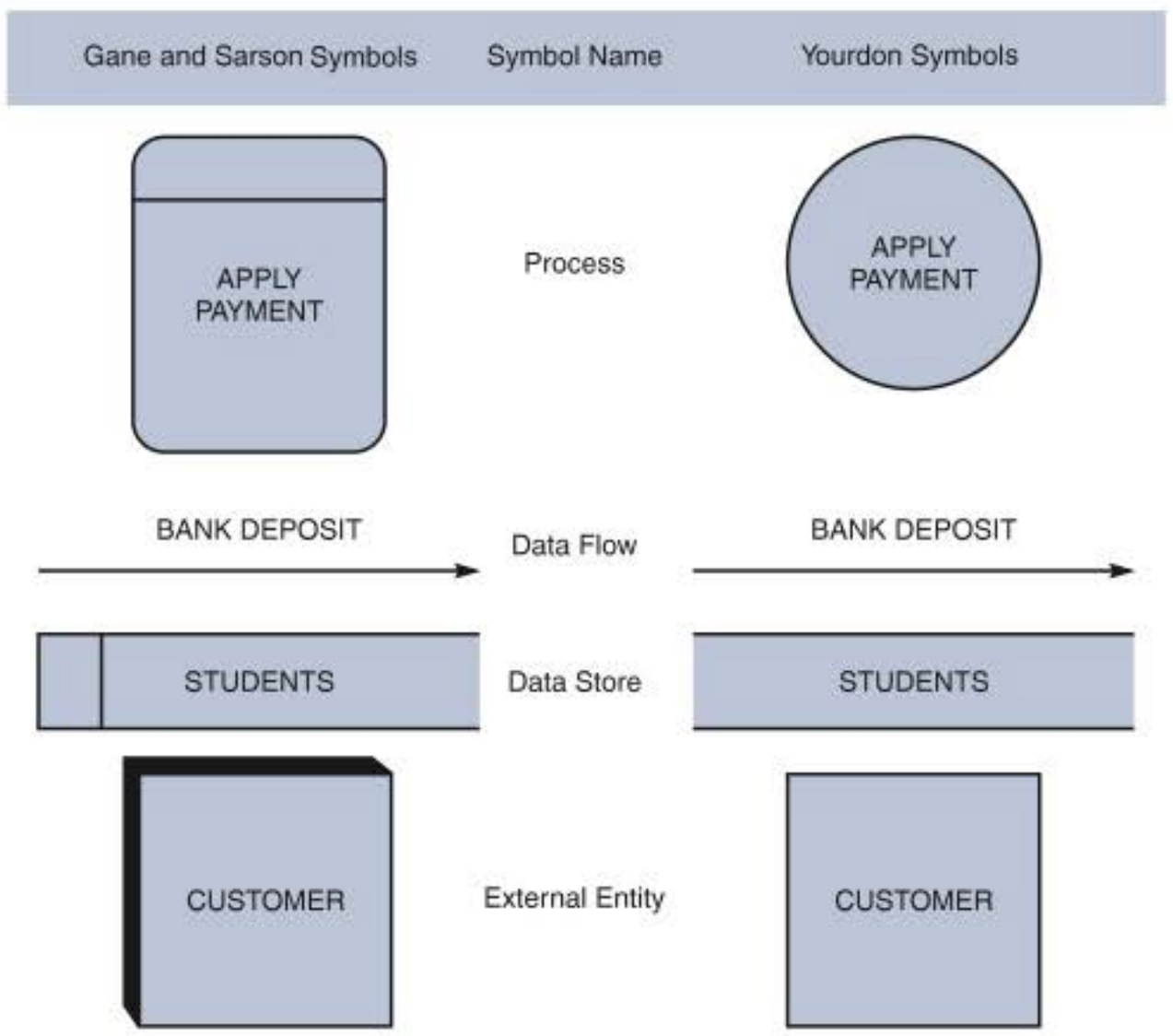


FIGURE 5-3 Data flow diagram symbols, symbol names, and examples of the Gane and Sarson and Yourdon symbol sets.

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 and clutter. 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 pictures.

FIGURE 5-4 Networks use various devices that act like *black boxes*. Cables carry data in and out, but internal operations are hidden inside the case.
© INSADCO Photography/Alamy

The network router shown in Figure 5-4 is an example of a black box. An observer can see cables that carry data into and out of the router, but the router's internal operations are not revealed — only the results are apparent.

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). Although the DFD does not show the detailed contents of a data flow, that information is included in the data dictionary, which is described later in this chapter.

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. Exceptions to the singular name rule are data flow names, such as GRADING PARAMETERS, where a singular name could mislead you into thinking a single parameter or single item of data exists.

Figure 5-5 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.

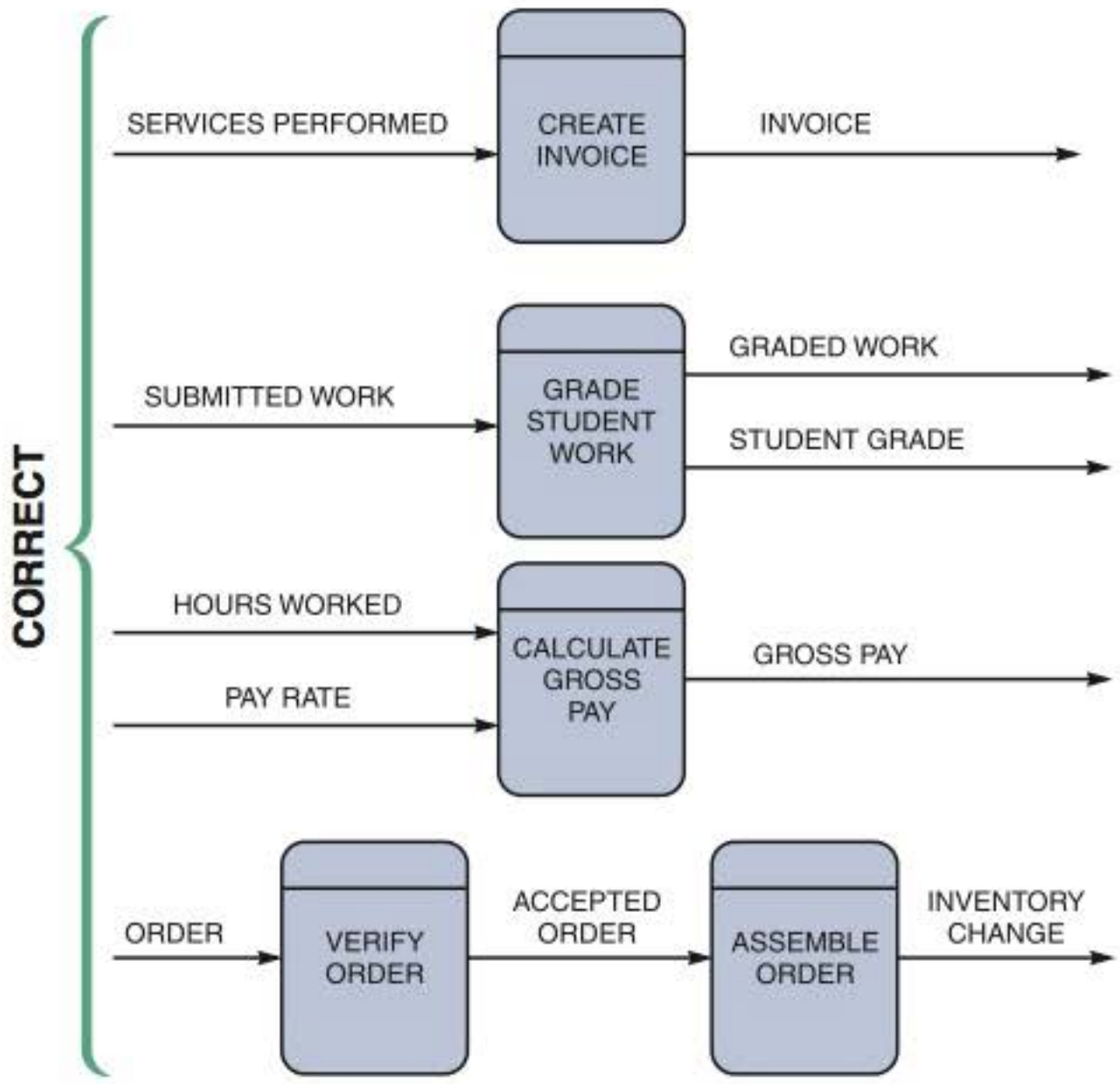


FIGURE 5-5 Examples of correct combinations of data flow and process symbols.
© Cengage Learning 2014

A process symbol can have more than one outgoing data flow, as shown in the GRADE 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 in Figure 5-5. A data flow, therefore, *must* have a process symbol on at least one end.

Figure 5-6 shows three data flow and process combinations that you must avoid:

- **Spontaneous generation.** The APPLY INSURANCE PREMIUM process, for instance, produces output, but has no input data flow. Because it has no input, the process is called a spontaneous generation process.
- **Black hole.** CALCULATE GROSS PAY is called a black hole process, which is a process that has input, but produces no output.
- **Gray hole.** A gray hole is a process that has at least one input and one output, but the input obviously is insufficient to generate the output shown. For example, a date of birth input is not sufficient to produce a final grade output in the CALCULATE GRADE process.

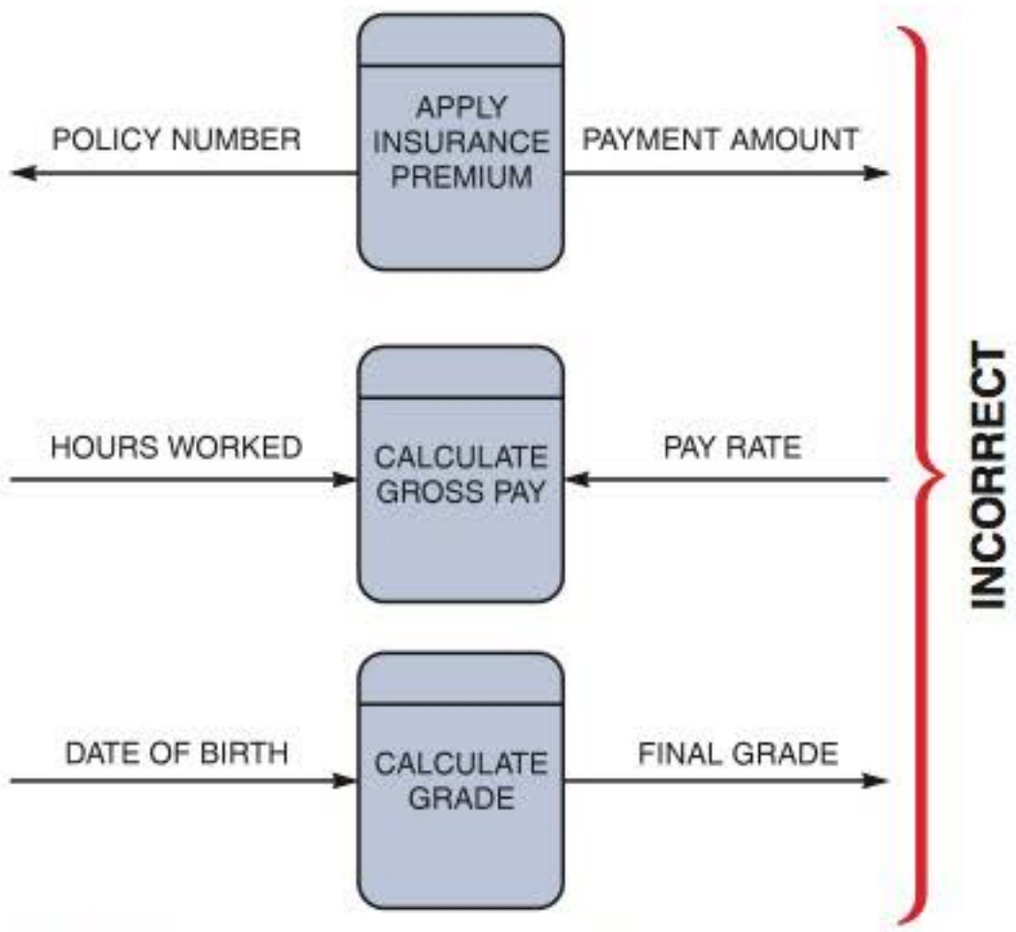


FIGURE 5-6 Examples of incorrect combinations of data flow and process symbols. APPLY INSURANCE PREMIUM has no input and is called a spontaneous generation process. CALCULATE GROSS PAY has no outputs and is called a black hole process. CALCULATE GRADE has an input that is obviously unable to produce the output. This process is called a gray hole.

Spontaneous generation, black holes, and gray holes are impossible logically in a DFD because a process must act on input, shown by an incoming data flow, and produce output, represented by an outgoing data flow.

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. Similarly, a company stores employee salary and deduction data during the year in order to print W-2 forms with

total earnings and deductions at the end of the year. A DFD does not show the detailed contents of a data store — the specific structure and data elements are defined in the data dictionary, which is discussed later in this chapter.

The physical characteristics of a data store are unimportant because you are concerned only with a logical model. Also, the length of time that the data is stored is unimportant — it can be a matter of seconds while a transaction is processed or a period of months while data is accumulated for year-end processing. What is important is that a process needs access to the data at some later time.

In a DFD, the Gane and Sarson 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. 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.

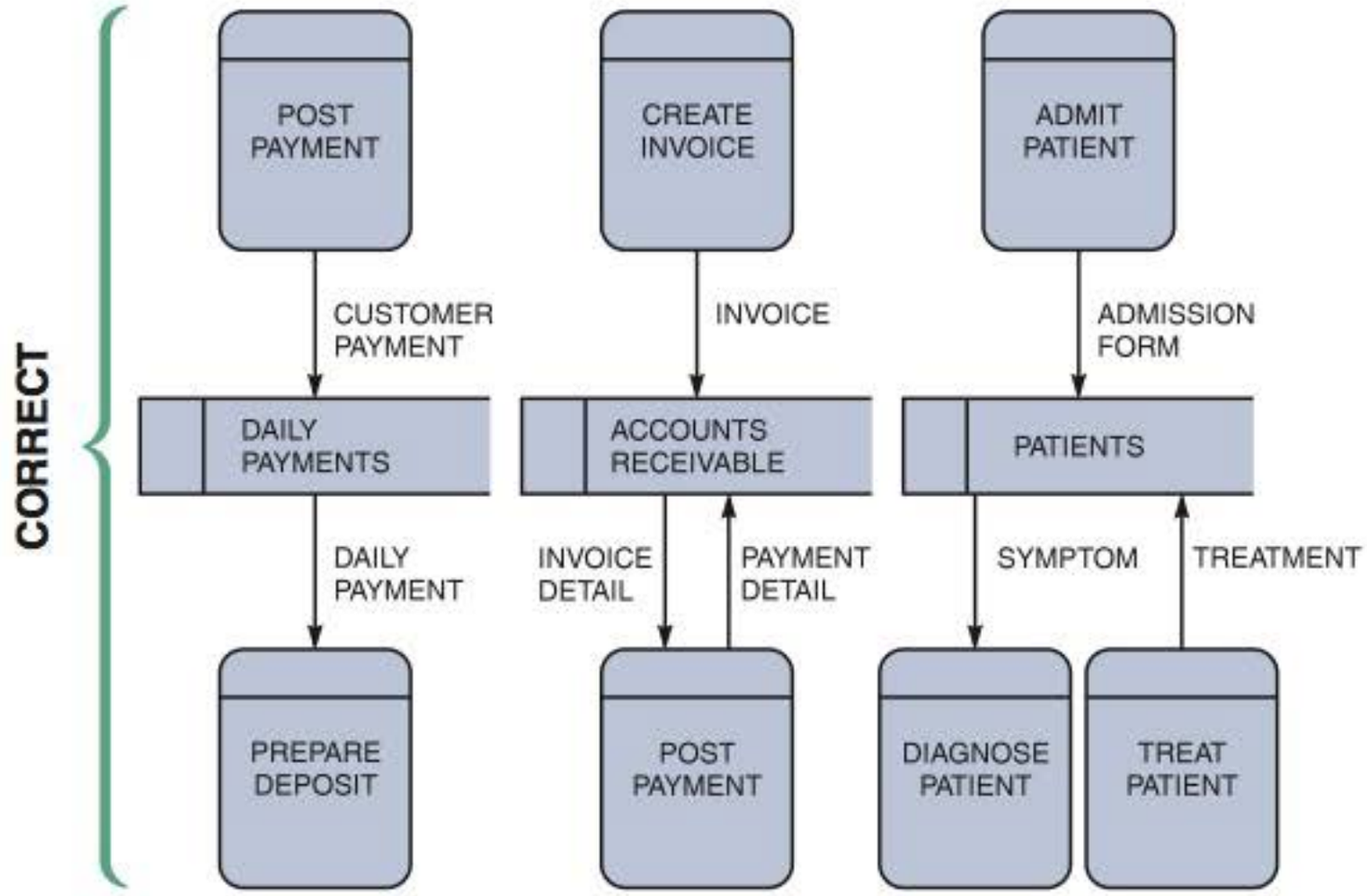


FIGURE 5-7 Examples of correct uses of data store symbols in a data flow diagram.

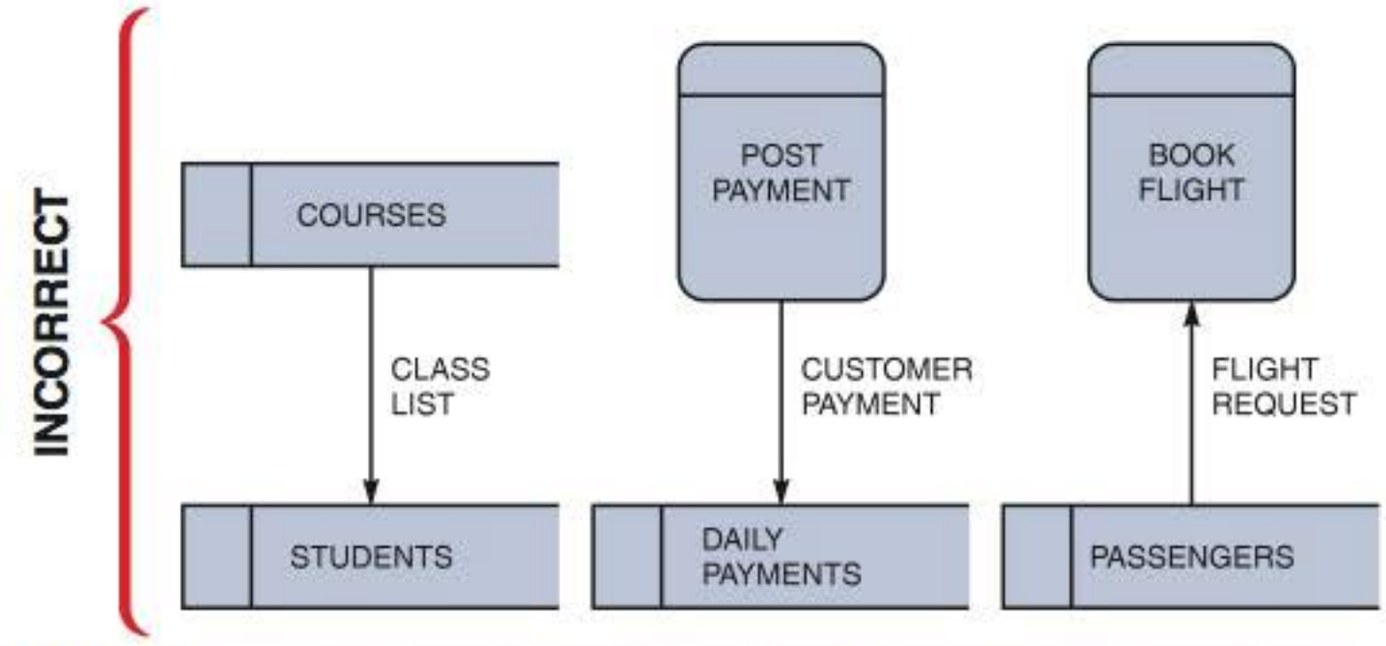


FIGURE 5-8 Examples of incorrect uses of data store symbols: Two data stores cannot be connected by a data flow without an intervening process, and each data store should have an outgoing and incoming data flow.

A data store must be connected to a process with a data flow. Figure 5-7 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.

Violations of the rule that a data store must have at least one incoming and one outgoing data flow are shown in Figure 5-8. In the first example, two data stores are connected incorrectly because no process is

between them. Also, COURSES has no incoming data flow and STUDENTS has no outgoing data flow. In the second and third examples, the data stores lack either an outgoing or incoming data flow.

There is an exception to the requirement that a data store must have at least one incoming and one outgoing data flow. In some situations, a data store has no input data flow because it contains fixed reference data that is not updated by the system. For example, consider a data store called TAX TABLE, which contains withholding tax data that a company downloads from the Internal Revenue Service. When the company runs its payroll, the CALCULATE WITHHOLDING process accesses data from this data store. On a DFD, this would be represented as a one-way outgoing data flow from the TAX TABLE data store into the CALCULATE WITHHOLDING process.

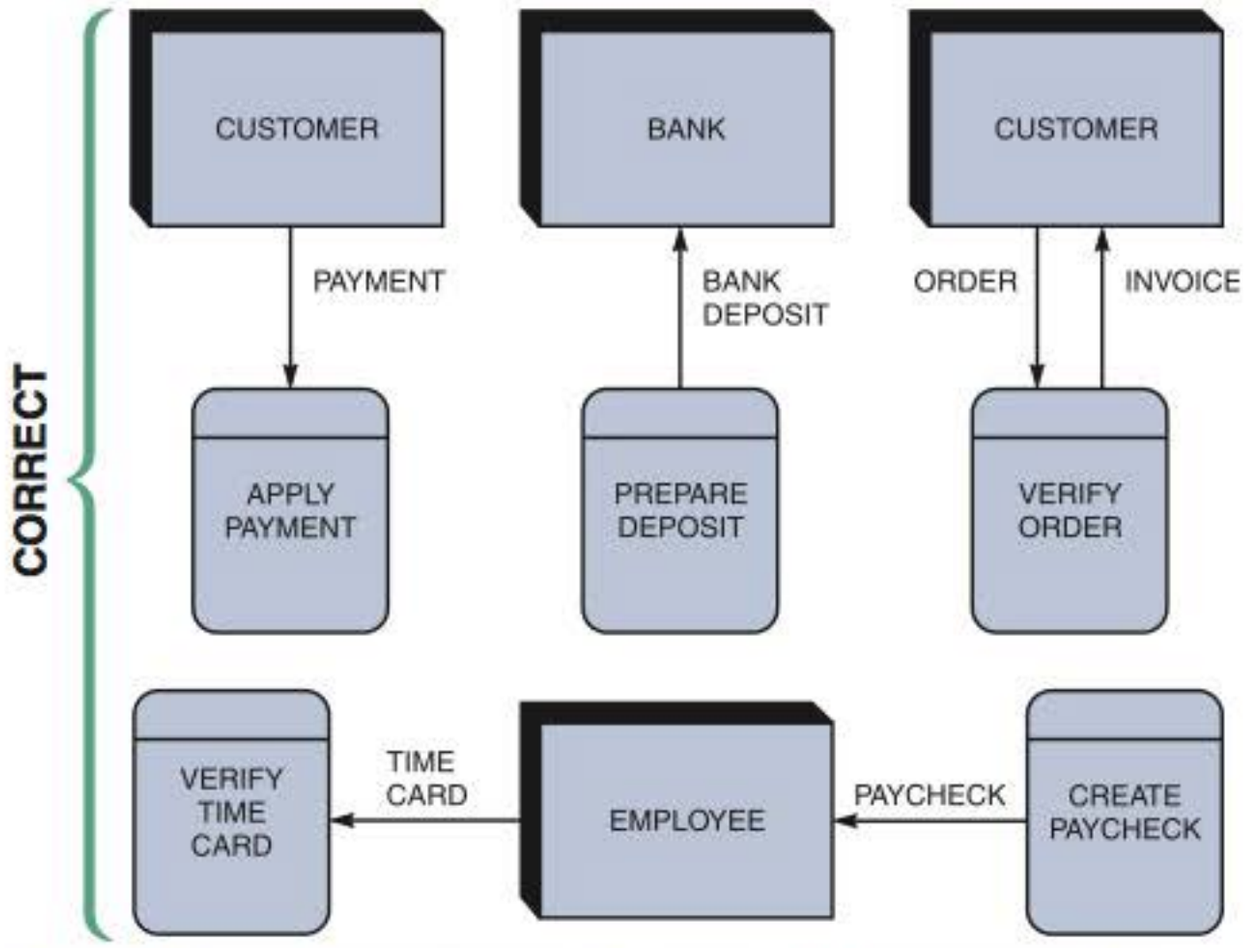


FIGURE 5-9 Examples of correct uses of external entities in a data flow diagram.

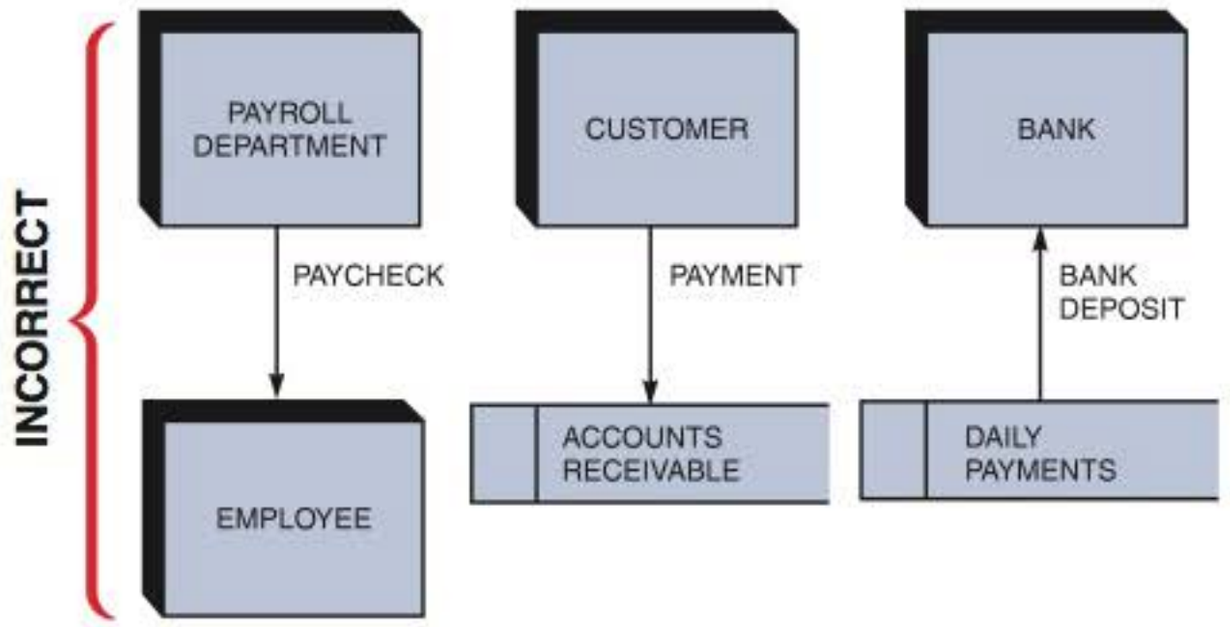


FIGURE 5-10 Examples of incorrect uses of external entities. An external entity must be connected by a data flow to a process, and not directly to a data store or to another external entity.

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 system, 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. Figures 5-9 and 5-10 show correct and incorrect examples of this rule.

With an understanding of the proper use of DFD symbols, you are ready to construct diagrams that use these symbols. Figure 5-11 on the next page shows a summary of the rules for using DFD symbols.

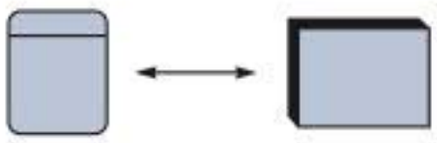
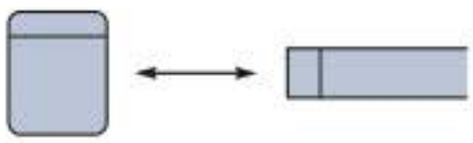
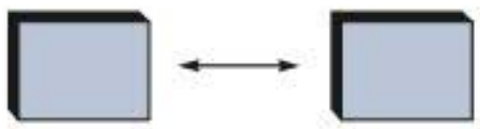
Correct and Incorrect Examples of Data Flows		
	Process to Process	✓
	Process to External Entity	✓
	Process to Data Store	✓
	External Entity to External Entity	✗
	External Entity to Data Store	✗
	Data Store to Data Store	✗

FIGURE 5-11 Examples of correct and incorrect uses of data flows.

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.

To learn how to construct DFDs, you will use examples of two information systems. The first example is a grading system that instructors use to assign final grades based on the scores that students receive during the term. The second example is an order system that a company uses to enter orders and apply payments against a customer's balance. First, you will review a set of guidelines for drawing DFDs. Then you will learn how to apply these guidelines and create a set of DFDs using a three-step process.

Video Learning Session
DFD Context Diagrams

If you have an MIS CourseMate access code, you can launch interactive Video Learning Sessions to help you understand systems development concepts and practice your skills. You can watch the sessions on your computer or mobile device, and pause, rewind, or replay a video at any time. To log on to the MIS CourseMate site at **www.cengagebrain.com**, you must create a student account and then register this book.

This session is about DFD context diagrams, why they are important, how to construct a context diagram, and how you can use a CASE tool to create a context diagram.

© craftvision/Stockphoto

Guidelines for Drawing DFDs

When you draw a context diagram and other DFDs, you should follow several 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. For example, the process name in Figure 5-12 is GRADING SYSTEM. Notice that the process name is the same as the system name. This is because

the context diagram shows the entire information system as if it were a single process. For processes in lower-level DFDs, you would use a verb followed by a descriptive noun, such as ESTABLISH GRADEBOOK, ASSIGN FINAL GRADE, or PRODUCE GRADE REPORT.

- Use unique names within each set of symbols. For instance, the diagram in Figure 5-12 shows only one entity named STUDENT and only one data flow named FINAL GRADE. Whenever you see the entity STUDENT on any other DFD in the grading system, you know that you are dealing with the same entity. Whenever the FINAL GRADE data flow appears, you know that you are dealing with the same data flow. The naming convention also applies to data stores.
- Do not cross lines. One way to achieve that goal is to restrict the number of symbols in any DFD. 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. A special notation, such as an asterisk, next to the symbol name and inside the duplicated symbols signifies that they are duplicated on the diagram.
- Provide a unique name and reference number for each process. 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.
- 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.

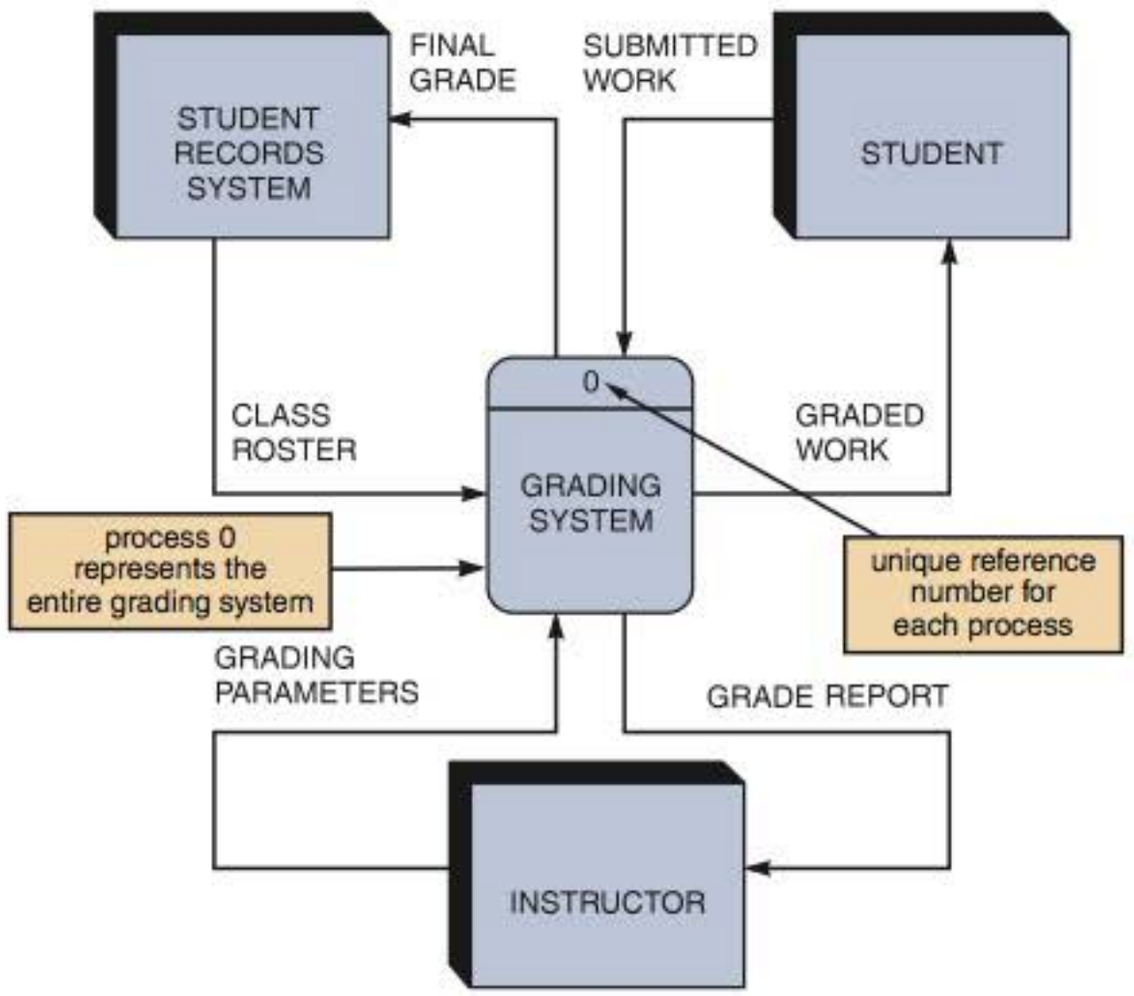


FIGURE 5-12 Context diagram DFD for a grading system.

© Cengage Learning 2014

Step 1: Draw a Context Diagram

The first step in constructing a set of DFDs is to draw a context diagram. A **context diagram** is a top-level view of an information system that shows the system's boundaries and scope. To draw a context diagram, you start by placing a single process symbol in the center of the page. The symbol represents the entire information system, and you identify it as **process 0** (the numeral zero, and not the letter O). Then you place the system entities around the perimeter of the page and use data flows to connect the entities to the central process. Data stores are not shown in the context diagram because they are contained within the system and remain hidden until more detailed diagrams are created.

How do you know which entities and data flows to place in the context diagram? You begin by reviewing the system requirements to identify all external data sources and destinations. During that process, you identify the entities, the name and content of the data flows, and the direction of the data flows. If you do that carefully, and you did a good job of fact-finding in the previous stage, you should have no difficulty drawing the context diagram. Now review the following context diagram examples.

EXAMPLE: CONTEXT DIAGRAM FOR A GRADING SYSTEM The context diagram for a grading system is shown in Figure 5-12. 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: CONTEXT DIAGRAM FOR AN ORDER SYSTEM The context diagram for an order system is shown in Figure 5-13. Notice that the ORDER SYSTEM process is at the center of the diagram and five entities surround the process. Three of the

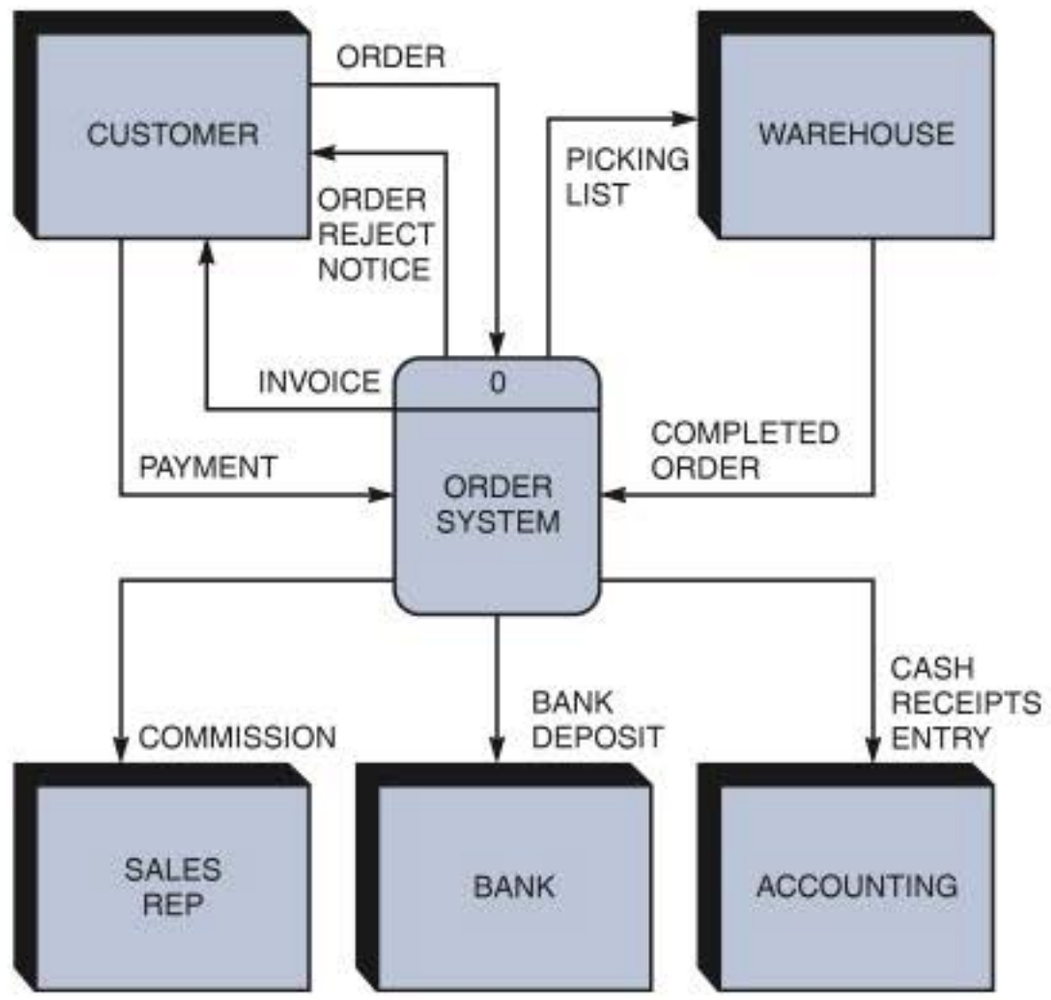


FIGURE 5-13 Context diagram DFD for an order system.

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 for the order system appears more complex than the grading system because it 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.

Video Learning Session
DFD Diagram 0

If you have an MIS CourseMate access code, you can launch interactive Video Learning Sessions to help you understand systems development concepts and practice your skills. You can watch the sessions on your computer or mobile device, and pause, rewind, or replay a video at any time. To log on to the MIS CourseMate site at **www.cengagebrain.com**, you must create a student account and then register this book.

This session is about DFD diagram 0. You'll learn what a DFD diagram 0 is, how to create a diagram 0, and how you can use a CASE tool to create a DFD diagram 0.

© craftvision/iStockphoto

Step 2: Draw a Diagram 0 DFD

In the previous step, you learned that a context diagram provides the most general view of an information system and contains a single process symbol, which is like a black box. To show the detail inside the black box, you create DFD diagram 0. **Diagram 0** (the numeral zero, and not the letter O) zooms in on the system and shows major internal processes, data flows, and data stores. Diagram 0 also repeats the entities and data flows that appear in the context diagram. When you expand the context diagram into DFD diagram 0, you must retain all the connections that flow into and out of process 0.

The real-life scene in Figure 5-14 represents a complex manufacturing system with many interactive processes and data. In a large system such as this, each process in diagram 0 could represent a separate system such as inventory, production control, and scheduling. Diagram 0 provides an overview of all the components that interact to form the overall system. Now review the following diagram 0 examples.

FIGURE 5-14 Complex manufacturing systems require many interactive processes and data sources.

© gerenme/iStockphoto

EXAMPLE: DIAGRAM 0 DFD FOR A GRADING SYSTEM Figure 5-15 on the next page shows a context diagram at the top and diagram 0 beneath it. Notice that diagram 0 is an expansion of process 0. Also notice that the three same entities (STUDENT RECORDS SYSTEM, STUDENT, and INSTRUCTOR) and the same six data flows (FINAL GRADE,

Context Diagram for Grading System

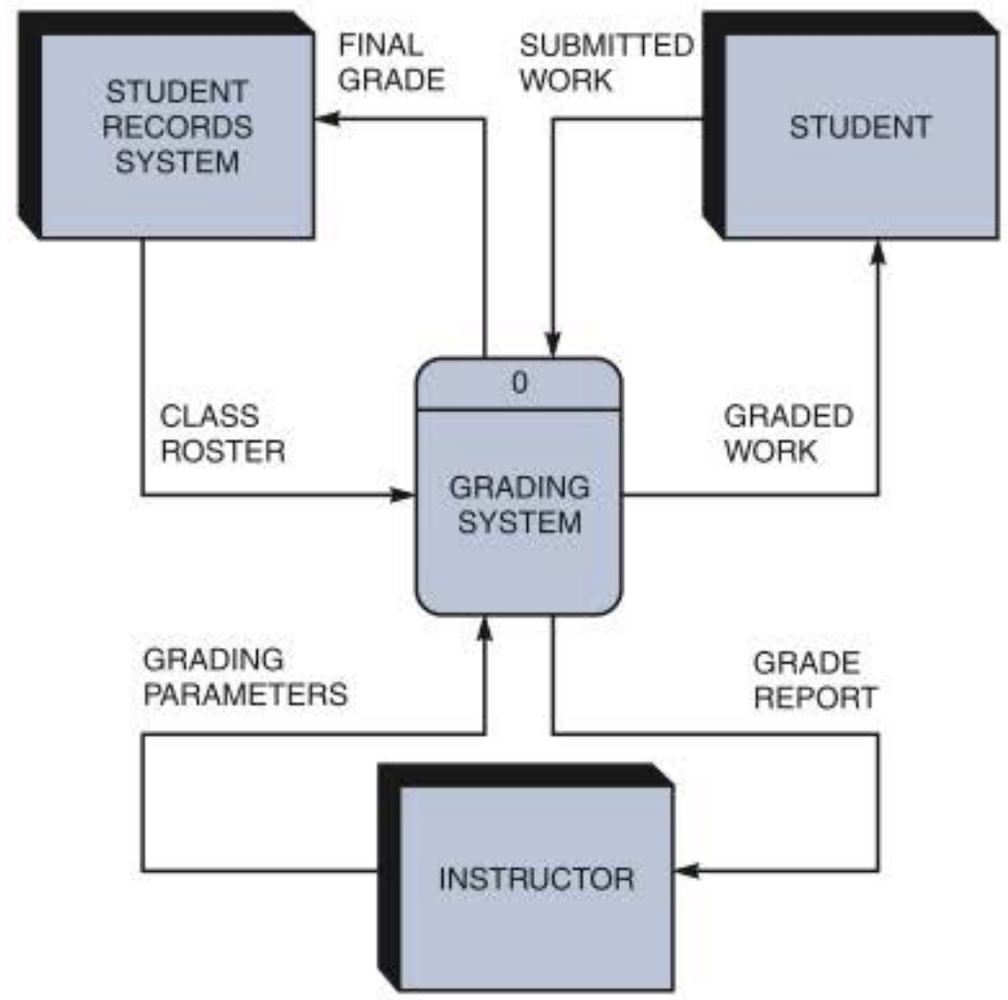


Diagram 0 for Grading System

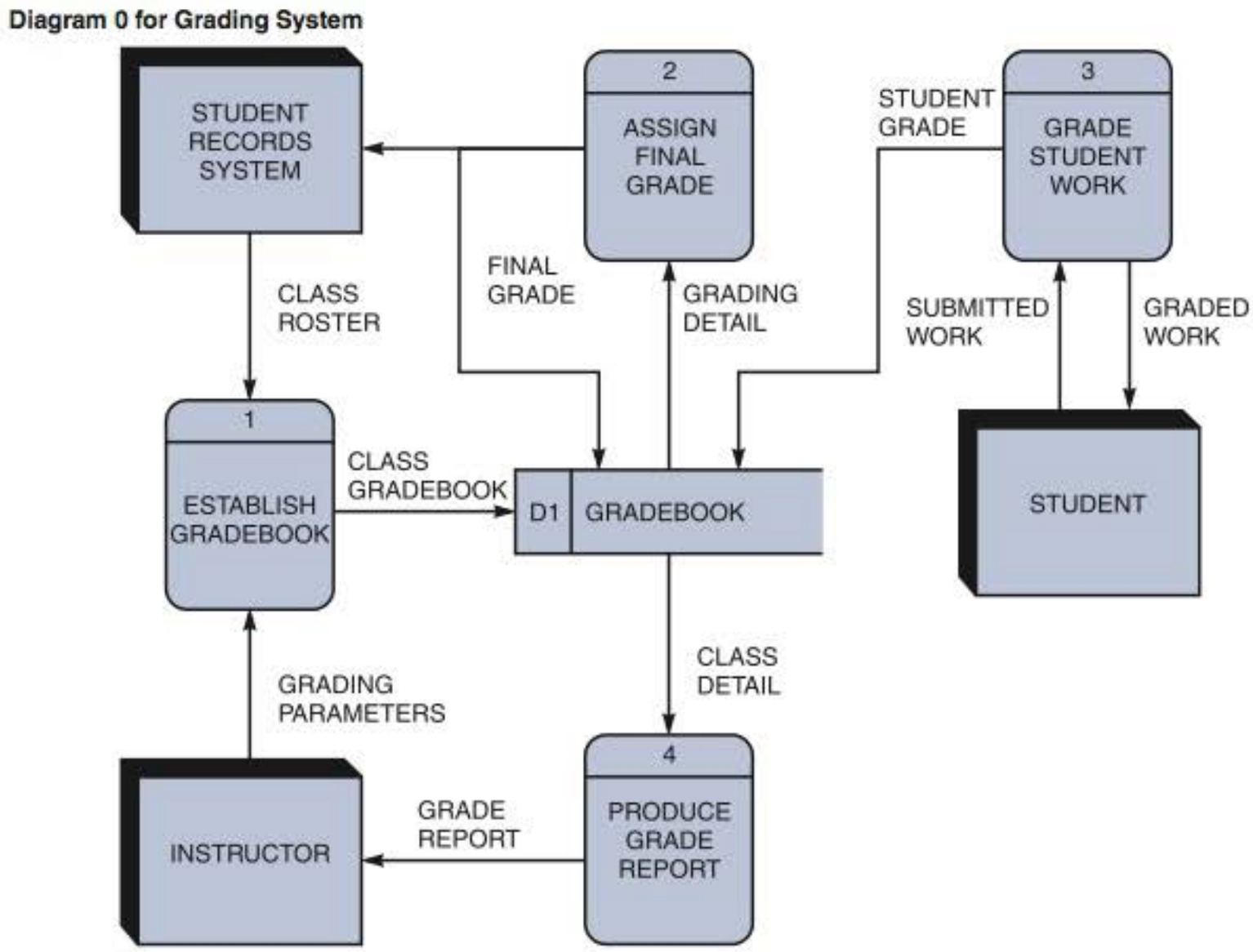


FIGURE 5-15 Context diagram and diagram 0 for the grading system.

CLASS ROSTER, SUBMITTED WORK, GRADED WORK, GRADING PARAMETERS, and GRADE REPORT) appear in both diagrams. In addition, diagram 0 expands process 0 to reveal four internal processes, one data store, and five additional data flows.

Notice that each process in diagram 0 has a reference number: ESTABLISH GRADEBOOK is 1, ASSIGN FINAL GRADE is 2, GRADE STUDENT WORK is 3, and PRODUCE GRADE REPORT is 4. These reference numbers are important because they identify a series of DFDs. If more detail were needed for ESTABLISH GRADEBOOK, for example, you would draw a diagram 1 because ESTABLISH GRADEBOOK is process 1.

The process numbers do not suggest that the processes are accomplished in a sequential order. Each process always is considered to be available, active, and awaiting data to be processed. If processes must be performed in a specific sequence, you document the information in the process descriptions (discussed later in this chapter), not in the DFD.

The FINAL GRADE data flow output from the ASSIGN FINAL GRADE process is a diverging data flow that becomes an input to the STUDENT RECORDS SYSTEM entity and to the GRADEBOOK data store. A **diverging data flow** is a data flow in which the same data travels to two or more different locations. In that situation, a diverging data flow is the best way to show the flow rather than showing two identical data flows, which could be misleading.

If the same data flows in both directions, you can use a double-headed arrow to connect the symbols. To identify specific data flows into and out of a symbol, however, you use separate data flow symbols with single arrowheads. For example, in Figure 5-15, the separate data flows (SUBMITTED WORK and GRADED WORK) go into and out of the GRADE STUDENT WORK process.

Because diagram 0 is an exploded version of process 0, it shows considerably more detail than the context diagram. You also can refer to diagram 0 as a partitioned or decomposed view of process 0. When you explode a DFD, the higher-level diagram is called the **parent diagram**, and the lower-level diagram is referred to as the **child diagram**. The grading system is simple enough that you do not need any additional DFDs to model the system. At that point, the four processes, the one data store, and the 10 data flows can be documented in the data dictionary.

When you create a set of DFDs for a system, you break the processing logic down into smaller units, called functional primitives, that programmers will use to develop code. A **functional primitive** is a process that consists of a single function that is not exploded further. For example, each of the four processes shown in the lower portion of Figure 5-15 is a functional primitive. You document the logic for a functional primitive by writing a process description in the data dictionary. Later, when the logical design is implemented as a physical system, programmers will transform each functional primitive into program code and modules that carry out the required steps. Deciding whether to explode a process further or determine that it is a functional primitive is a matter of experience, judgment, and interaction with programmers who must translate the logical design into code.

EXAMPLE: DIAGRAM 0 DFD FOR AN ORDER SYSTEM Figure 5-16 on the next page shows the diagram 0 for an order system. Process 0 on the order system's context diagram is exploded to reveal three processes (FILL ORDER, CREATE INVOICE, and APPLY PAYMENT), one data store (ACCOUNTS RECEIVABLE), two additional data flows (INVOICE DETAIL and PAYMENT DETAIL), and one diverging data flow (INVOICE).

The following walkthrough explains the DFD shown in Figure 5-16:

1. A CUSTOMER submits an ORDER. Depending on the processing logic, the FILL ORDER process either sends an ORDER REJECT NOTICE back to the customer or sends a PICKING LIST to the WAREHOUSE.
2. A COMPLETED ORDER from the WAREHOUSE is input to the CREATE INVOICE process, which outputs an INVOICE to both the CUSTOMER process and the ACCOUNTS RECEIVABLE data store.

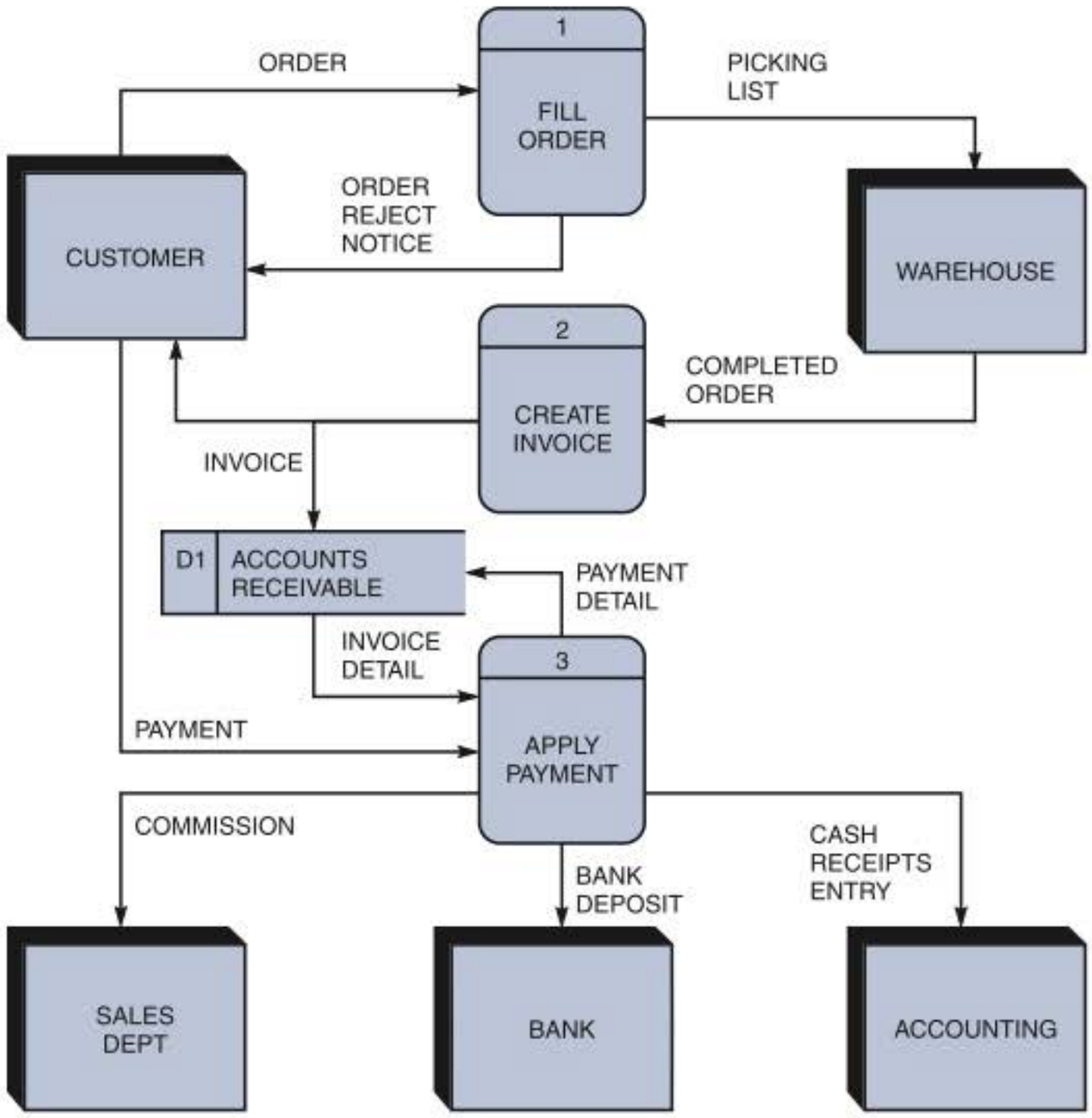


FIGURE 5-16 Diagram 0 DFD for the order system.

3. A CUSTOMER makes a PAYMENT that is processed by APPLY PAYMENT. APPLY PAYMENT requires INVOICE DETAIL input from the ACCOUNTS RECEIVABLE data store along with the PAYMENT. APPLY PAYMENT also outputs PAYMENT DETAIL back to the ACCOUNTS RECEIVABLE data store and outputs COMMISSION to the SALES DEPT, BANK DEPOSIT to the BANK, and CASH RECEIPTS ENTRY to ACCOUNTING.

The walkthrough of diagram 0 illustrates the basic requirements of the order system. To learn more, you would examine the detailed description of each separate process.

Step 3: Draw the Lower-Level Diagrams

This set of lower-level DFDs is based on the order system. To create lower-level diagrams, you must use leveling and balancing techniques. **Leveling** is the process of drawing a series of increasingly detailed diagrams, until all functional primitives are identified. **Balancing** maintains consistency among a set of DFDs by ensuring that input and output data flows align properly. Leveling and balancing are described in more detail in the following sections.

LEVELING EXAMPLES Leveling uses a series of increasingly detailed DFDs to describe an information system. For example, a system might consist of dozens, or even hundreds, of separate processes. Using leveling, an analyst starts with an overall view, which is a context diagram with a single process symbol. Next, the analyst creates diagram 0, which shows more detail. The analyst continues to create lower-level

DFDs until all processes are identified as functional primitives, which represent single processing functions. More complex systems have more processes, and analysts must work through many levels to identify the functional primitives. Leveling also is called **exploding, partitioning,** or **decomposing.**

Figures 5-16 and 5-17 provide an example of leveling. Figure 5-16 shows diagram 0 for an order system, with the FILL ORDER process labeled as process 1. Now consider Figure 5-17, which provides an exploded view of the FILL ORDER process. Notice that FILL ORDER (process 1) actually consists of three processes: VERIFY ORDER (process 1.1), PREPARE REJECT NOTICE (process 1.2), and ASSEMBLE ORDER (process 1.3).

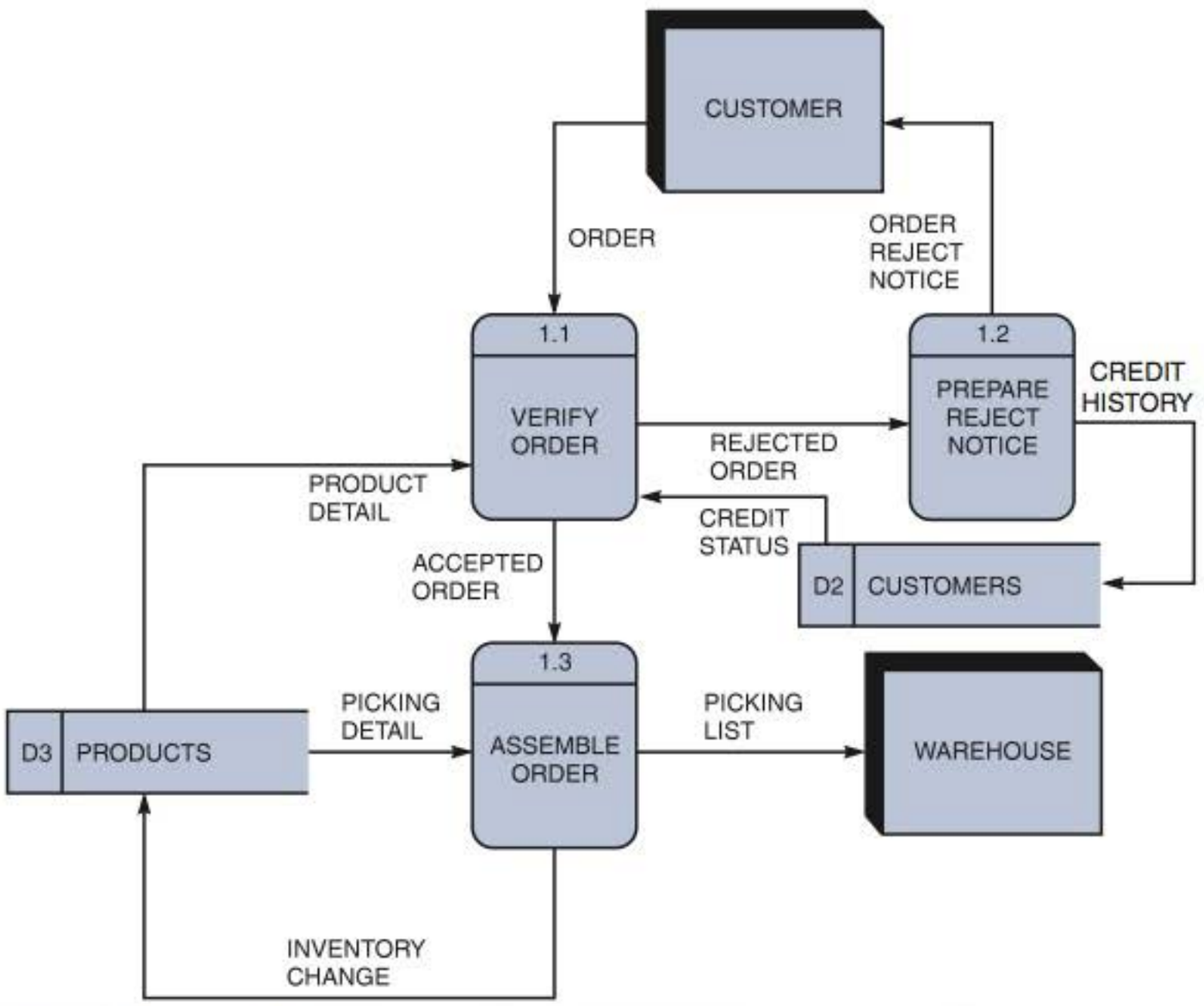


FIGURE 5-17 Diagram 1 DFD shows details of the FILL ORDER process in the order system.

As Figure 5-17 shows, all processes are numbered using a decimal notation consisting of the parent's reference number, a decimal point, and a sequence number within the new diagram. In Figure 5-17, the parent process of diagram 1 is process 1, so the processes in diagram 1 have reference numbers of 1.1, 1.2, and 1.3. If process 1.3, ASSEMBLE ORDER, is decomposed further, then it would appear in diagram 1.3 and the processes in diagram 1.3 would be numbered as 1.3.1, 1.3.2, 1.3.3, and so on. This numbering technique makes it easy to integrate and identify all DFDs.

When you compare Figures 5-16 and 5-17, you will notice that Figure 5-17 (the exploded FILL ORDER process) shows two data stores (CUSTOMER and PRODUCTS) that do not appear on Figure 5-16, which is the parent DFD. Why not? The answer is based on a simple rule: When drawing DFDs, you show a data store only

when two or more processes use that data store. The CUSTOMER and PRODUCTS data stores were internal to the FILL ORDER process, so the analyst did not show them on diagram 0, which is the parent. When you explode the FILL ORDER process into diagram 1 DFD, however, you see that three processes (1.1, 1.2, and 1.3) interact with the two data stores, which now are shown.

Now compare Figure 5-17 (on the previous page) and Figure 5-18. Notice that Figure 5-18 shows the same data flows as Figure 5-17, but does not show the CUSTOMER and WAREHOUSE entities. Analysts often use this technique to simplify a DFD and reduce unnecessary clutter. Because the missing symbols appear on the parent DFD, you can refer to that diagram to identify the source or destination of the data flows.

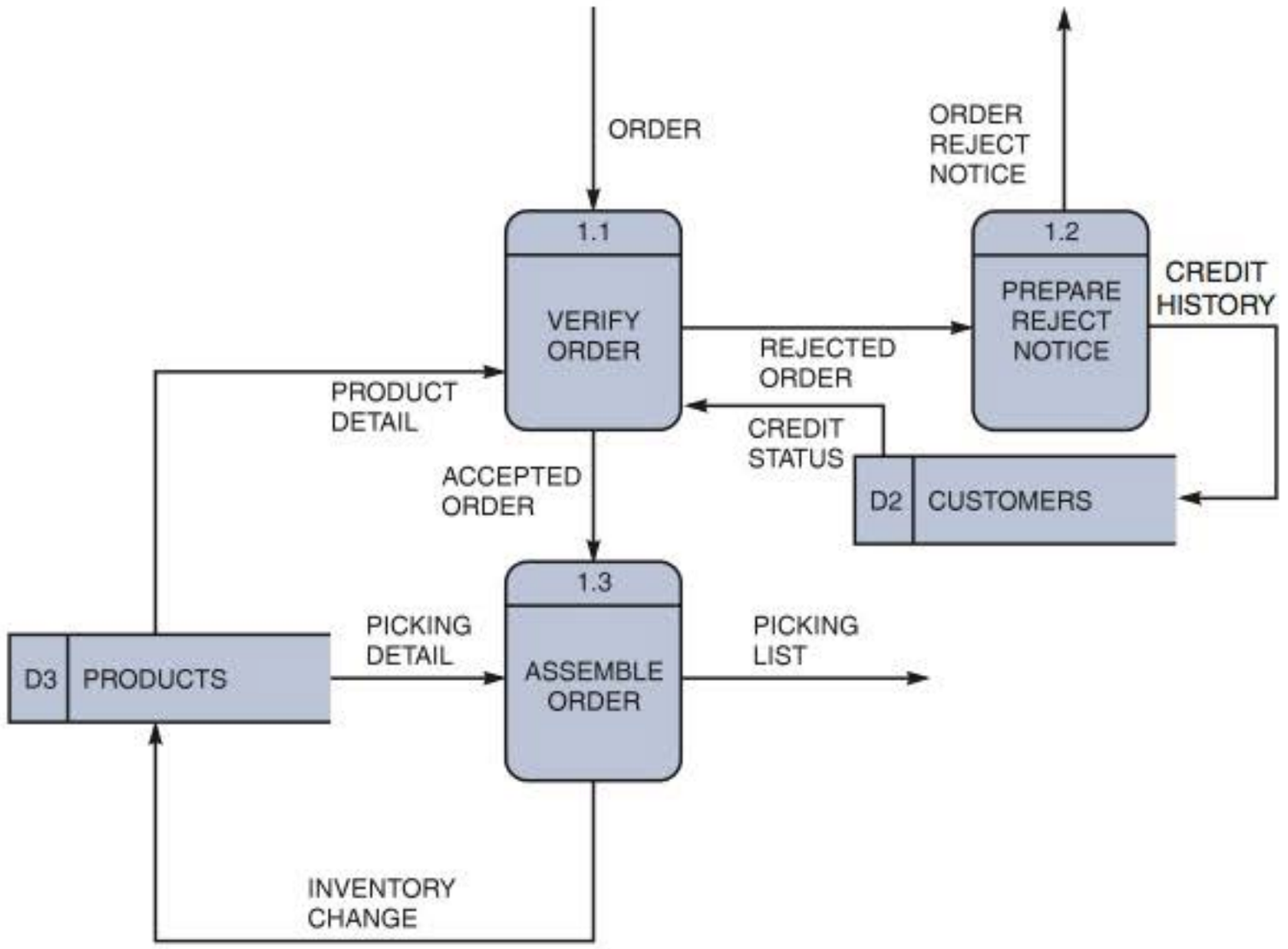
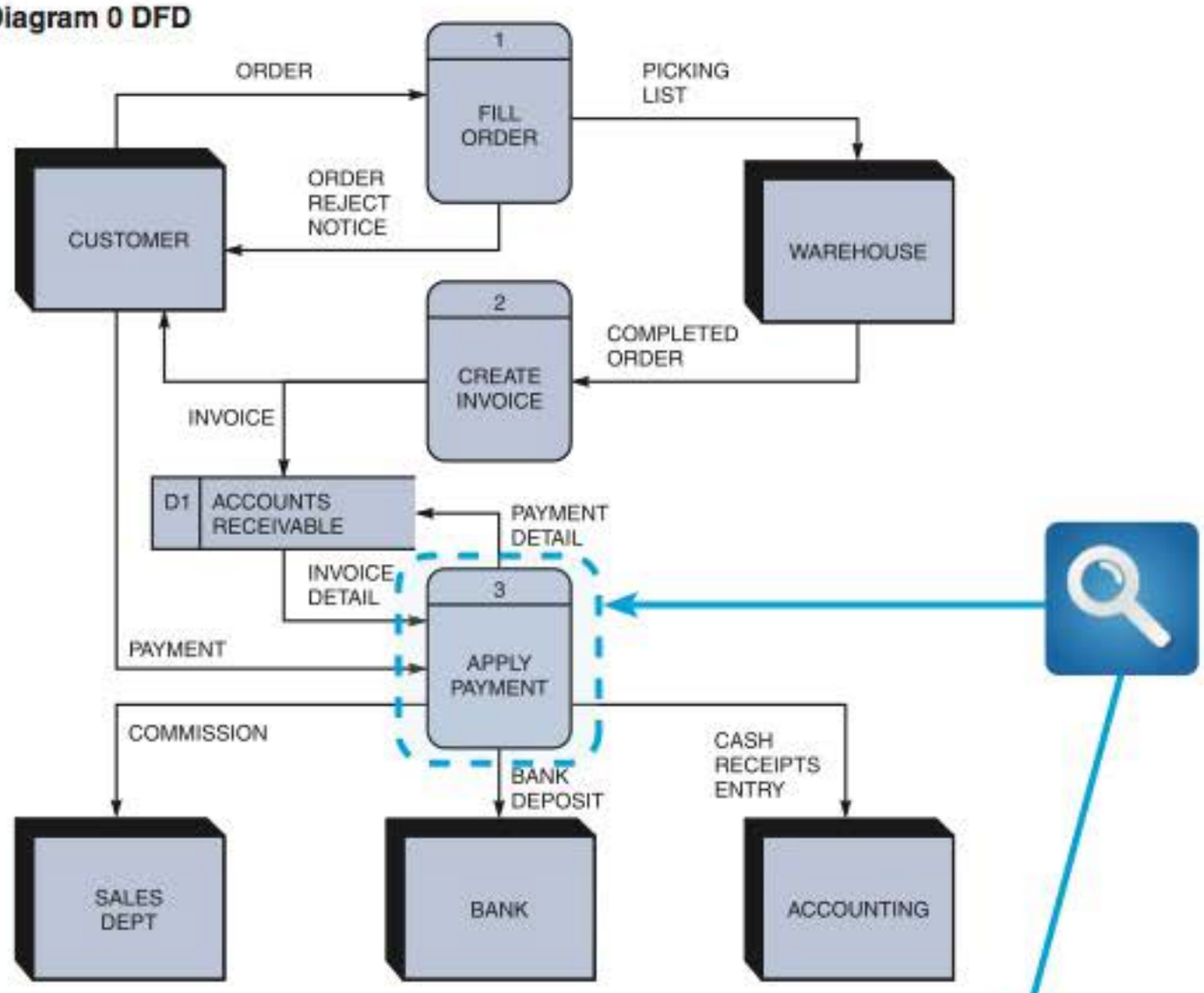


FIGURE 5-18 This diagram does not show the symbols that connect to data flows entering or leaving FILL ORDER on the context diagram.

BALANCING EXAMPLES Balancing ensures that the input and output data flows of the parent DFD are maintained on the child DFD. For example, Figure 5-19 shows two DFDs: The order system diagram 0 is shown at the top of the figure, and the exploded diagram 3 DFD is shown at the bottom.

The two DFDs are balanced because the child diagram at the bottom has the same input and output flows as the parent process 3 shown at the top. To verify the balancing, notice that the parent process 3, APPLY PAYMENT, has one incoming data flow from an external entity, and three outgoing data flows to external entities. Now examine the child DFD, which is diagram 3. Now, ignore the internal data flows and count the data flows to and from external entities. You will see that the three processes maintain the same one incoming and three outgoing data flows as the parent process.

Order System Diagram 0 DFD



Order System Diagram 3 DFD

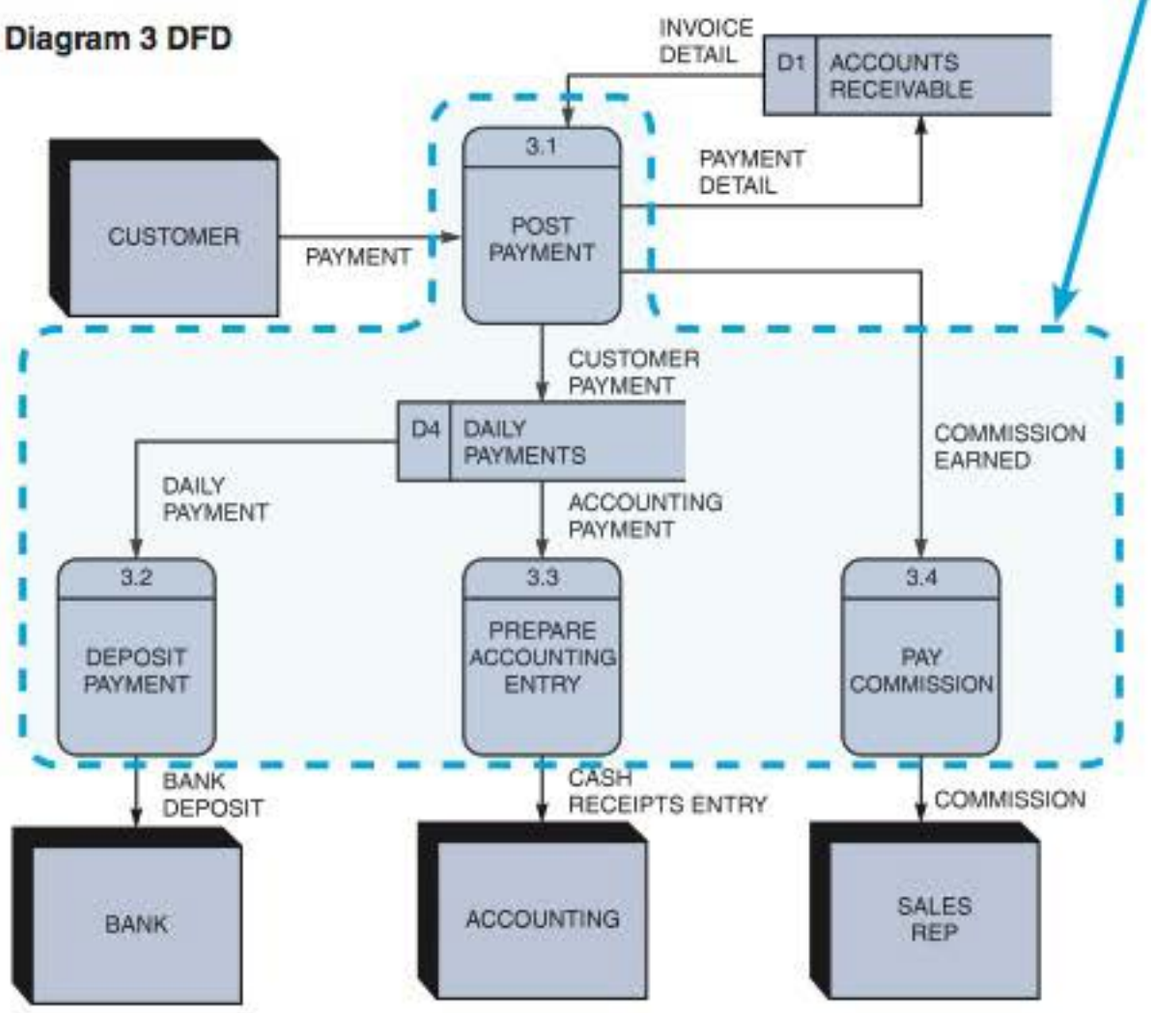


FIGURE 5-19 The order system diagram 0 is shown at the top of the figure, and exploded diagram 3 DFD (for the APPLY PAYMENT process) is shown at the bottom. The two DFDs are balanced because the child diagram at the bottom has the same input and output flows as the parent process 3 shown at the top.

Another example of balancing is shown in Figures 5-20 and 5-21 on the next page. The DFDs in these figures were created using Visible Analyst, a popular CASE tool.

Figure 5-20 shows a sample context diagram. The process 0 symbol has two input flows and two output flows. Notice that process 0 can be considered as a black box, with no internal detail shown. In Figure 5-21, process 0 (the parent DFD) is exploded into the next level of detail. Now three processes, two data stores, and four internal data flows are visible. Notice that the details of process 0 are shown inside a dashed line, just as if you could see inside the process.

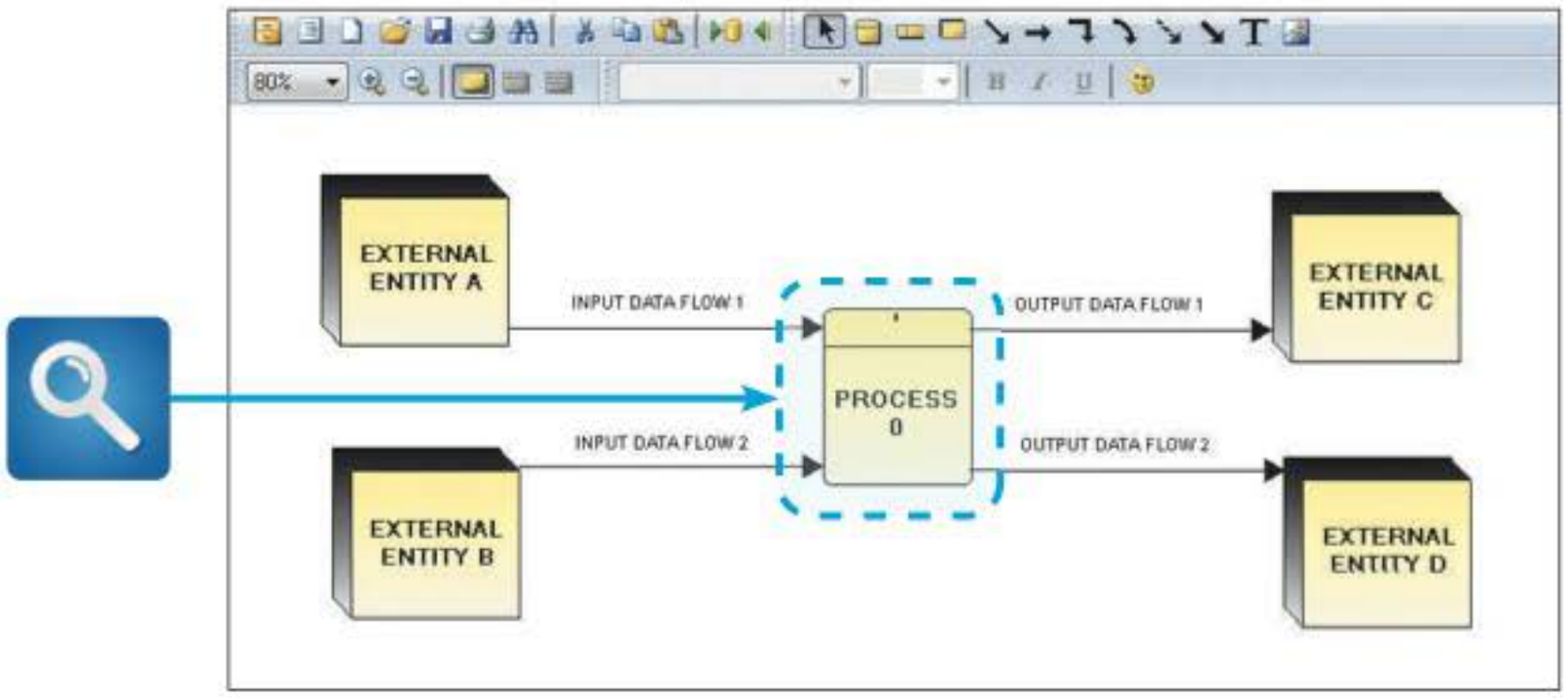


FIGURE 5-20 Example of a parent DFD diagram, showing process 0 as a black box.

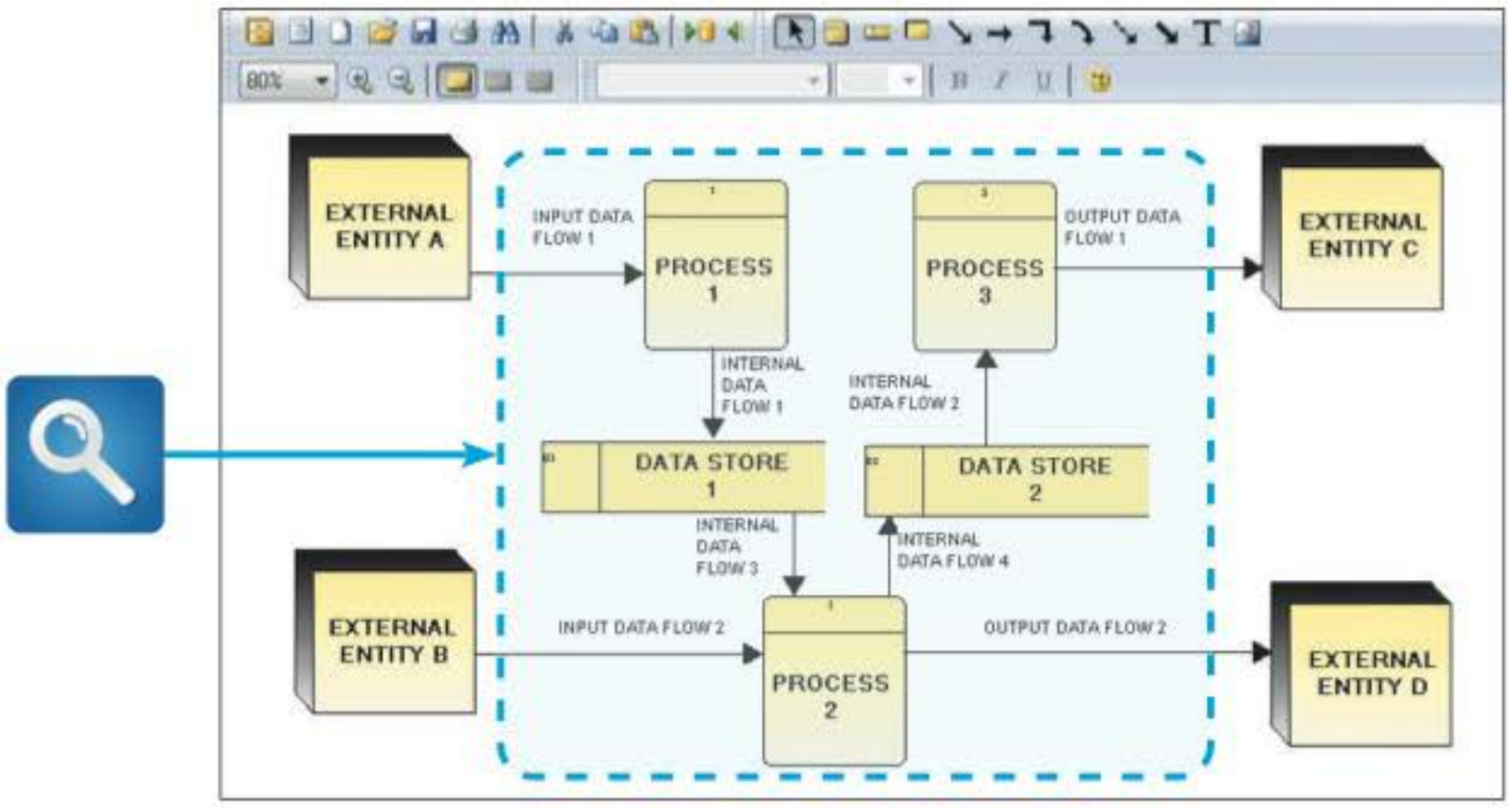


FIGURE 5-21 In the next level of detail, the process 0 black box reveals three processes, two data stores, and four internal data flows — all of which are shown inside the dashed line.

The DFDs in Figures 5-20 and 5-21 are balanced because the four data flows into and out of process 0 are maintained on the child DFD. The DFDs also are leveled because each internal process is numbered to show that it is a child of the parent process.

CASE IN POINT 5.1: BIG TEN UNIVERSITY

You are the IT director at Big Ten University. As part of a training program, you decide to draw a DFD that includes some obvious mistakes to see whether your newly hired junior analysts can find them. You came up with the diagram 0 DFD shown in Figure 5-22. Based on the rules explained in this chapter, how many problems should the analysts find?

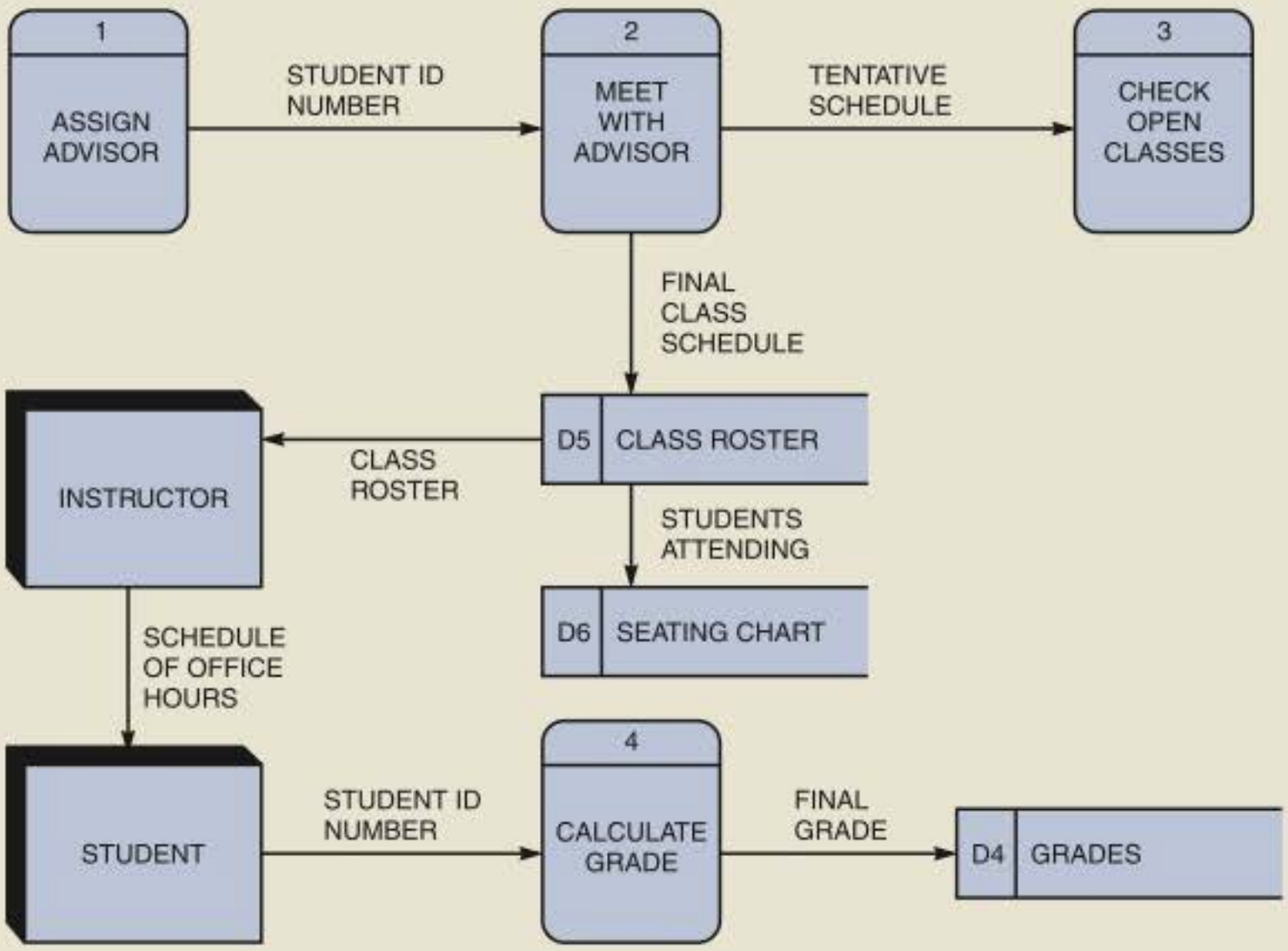


FIGURE 5-22 What are the mistakes in this diagram 0 DFD?

DATA DICTIONARY

A set of DFDs produces a logical model of the system, but the details within those DFDs are documented separately in a data dictionary, which is the second component of structured analysis.

A **data dictionary**, or **data repository**, is a central storehouse of information about the system's data. An analyst uses the data dictionary to collect, document, and organize specific facts about the system, including the contents of data flows, data stores, entities, and processes. The data dictionary also defines and describes all data elements and meaningful combinations of data elements. A **data element**, also called a **data item** or **field**, is the smallest piece of data that has meaning within an information system. Examples of data elements are student grade, salary, Social Security number, account balance, and company name. Data elements are combined into **records**, also called **data structures**. A record is a meaningful combination of related data elements that is included in a data flow or retained in a data store. For example, an auto parts store inventory record might include part number, description, supplier code, minimum and maximum stock levels, cost, and list price.

Significant relationships exist among the items in a data dictionary. For example, data stores and data flows are based on data structures, which in turn are composed of data elements. Data flows are connected to data stores, entities, and processes. Accurately documenting these relationships is essential so the data dictionary is consistent with the DFDs. You can use CASE software to help you document the design.

TOOLKIT TIME

The CASE tools in Part B of the Systems Analyst's Toolkit can help you document business functions and processes. To learn more about these tools, turn to Part B of the four-part Toolkit that follows Chapter 12.

Using CASE Tools for Documentation

The more complex the system, the more difficult it is to maintain full and accurate documentation. Fortunately, modern CASE tools simplify the task. For example, in the Visible Analyst CASE tool, documentation automatically flows from the modeling diagrams into the central repository, along with information entered by the user. This section contains several examples of Visible Analyst screens that show the data repository and its contents.

A CASE repository ensures data consistency, which is especially important where multiple systems require the same data. In a large company, for example, the sales, accounting, and shipping systems all might use a data element called CUSTOMER NUMBER. Once the CUSTOMER NUMBER element has been defined in the repository, it can be accessed by other processes, data flows, and data stores. The result is that all systems across the enterprise can share data that is up to date and consistent. You will learn more about CASE tools in Part B of the Systems Analyst's Toolkit.

Documenting the Data Elements

You must document every data element in the data dictionary. Some analysts like to record their notes on online or manual forms. Others prefer to enter the information directly into a CASE tool. Several of the DFDs and data dictionary entries that appear in this chapter were created using a popular CASE tool called Visible Analyst. Although other CASE tools might use other terms or display the information differently, the objective is the same: to provide clear, comprehensive information about the data and processes that make up the system.

Figure 5-23 shows how the analyst used an online documentation form to record information for the SOCIAL SECURITY NUMBER data element. Notice that the figure caption identifies eight specific characteristics for this data element.

1. Online or manual documentation entries often indicate which system is involved. This is not necessary with a CASE tool because all information is stored in one file that is named for the system.
2. The data element has a standard label that provides consistency throughout the data dictionary.
3. The data element can have an alternative name, or alias.
4. This entry indicates that the data element consists of nine numeric characters.
5. Depending on the data element, strict limits might be placed on acceptable values.
6. The data comes from the employee's job application.
7. This entry indicates that only the payroll department has authority to update or change this data.
8. This entry indicates the individual or department responsible for entering and changing data.

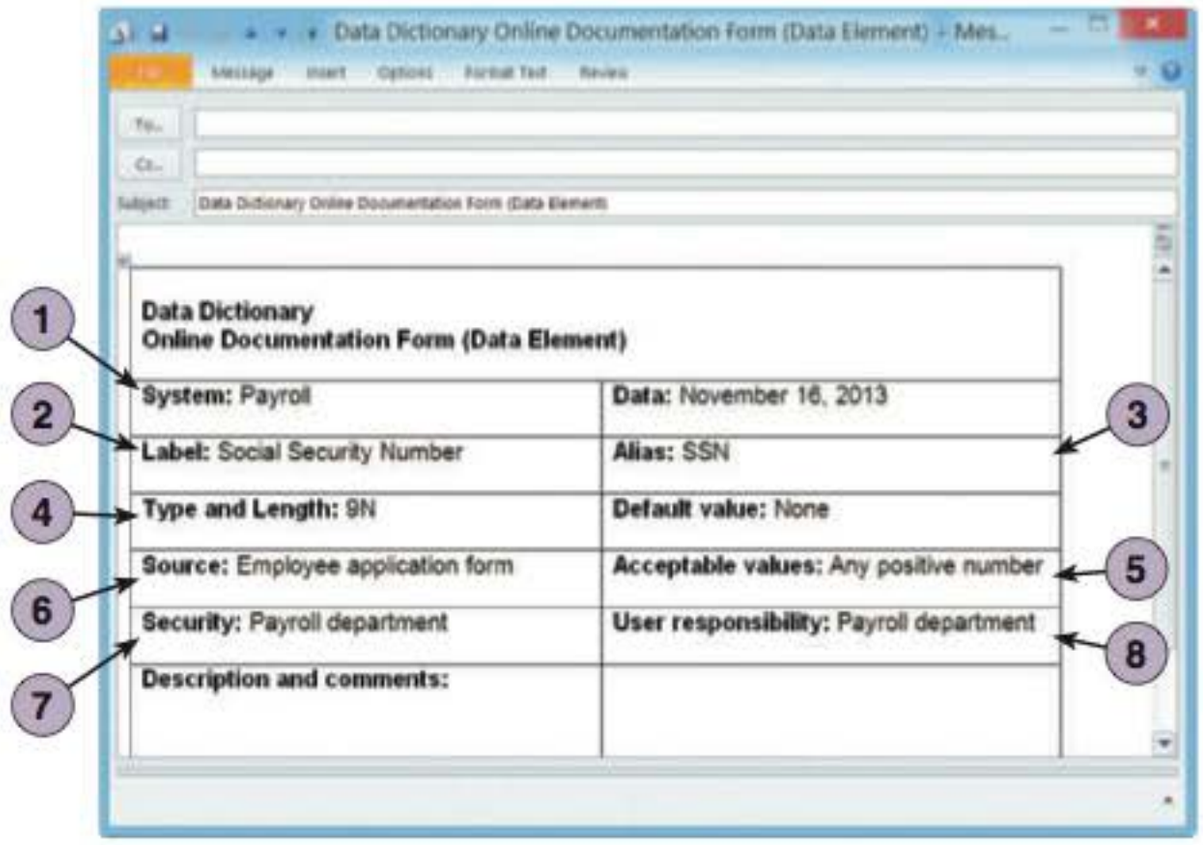


FIGURE 5-23 Using an online documentation form, the analyst has recorded information for a data element named SOCIAL SECURITY NUMBER. Later, the analyst will create a data dictionary entry using a CASE tool.

Figure 5-24 shows a sample screen that illustrates how the SOCIAL SECURITY NUMBER data element might be recorded in the Visible Analyst data dictionary.

Regardless of the terminology or method, the following attributes usually are recorded and described in the data dictionary:

Data element name or label. The data element's standard name, which should be meaningful to users.

Alias. Any name(s) other than the standard data element name; this alternate name is called an **alias**. For example, if you have a data element named CURRENT BALANCE, various users might refer to it by alternate names such as OUTSTANDING BALANCE, CUSTOMER BALANCE, RECEIVABLE BALANCE, or AMOUNT OWED.

Type and length. **Type** refers to whether the data element contains numeric, alphabetic, or character values. **Length** is the maximum number of characters for an alphabetic or character data element or the maximum number of digits and number of decimal positions for a numeric data element. In addition to text and numeric data, sounds and images also can be stored in digital form. In some systems, these binary data objects are managed and processed just as traditional data elements are. For example, an employee record might include a digitized photo image of the person.

FIGURE 5-24 A Visible Analyst screen describes the data element named SOCIAL SECURITY NUMBER. Notice that many of the items were entered from the online form shown in Figure 5-23.

Screenshot used with permission from Visible Systems Corporation.

Default value. The default value is the value for the data element if a value otherwise is not entered for it. For example, all new customers might have a default value of $500 for the CREDIT LIMIT data element.

Acceptable values. Specification of the data element's **domain,** which is the set of values permitted for the data element; these values either can be specifically listed or referenced in a table, or can be selected from a specified range of values. You also would indicate if a value for the data element is optional. Some data elements have additional **validity rules**. For example, an employee's salary must be within the range defined for the employee's job classification.

Source. The specification for the origination point for the data element's values. The source could be a specific form, a department or outside organization, another information system, or the result of a calculation.

Security. Identification for the individual or department that has access or update privileges for each data element. For example, only a credit manager has the authority to change a credit limit, while sales reps are authorized to access data in a read-only mode.

Responsible user(s). Identification of the user(s) responsible for entering and changing values for the data element.

Description and comments. This part of the documentation allows you to enter additional notes.

Documenting the Data Flows

In addition to documenting each data element, you must document all data flows in the data dictionary. Figure 5-25 shows a definition for a data flow named COMMISSION. The information on the manual form at the top was entered into the CASE tool data dictionary at the bottom of Figure 5-25.

Although terms can vary, the typical attributes are as follows:

Data flow name or label. The data flow name as it appears on the DFDs.

Description. Describes the data flow and its purpose.

Alternate name(s). Aliases for the DFD data flow name(s).

Origin. The DFD beginning, or source, for the data flow; the origin can be a process, a data store, or an entity.

Destination. The DFD ending point(s) for the data flow; the destination can be a process, a data store, or an entity.

Record. Each data flow represents a group of related data elements called a record or data structure. In most data dictionaries, records are defined separately from the data flows and data stores. When records are defined, more than one data flow or data store can use the same record, if necessary.

Volume and frequency. Describes the expected number of occurrences for the data flow per unit of time. For example, if a company has 300 employees, a TIME CARD data flow would involve 300 transactions and records each week as employees submit their work hour data.

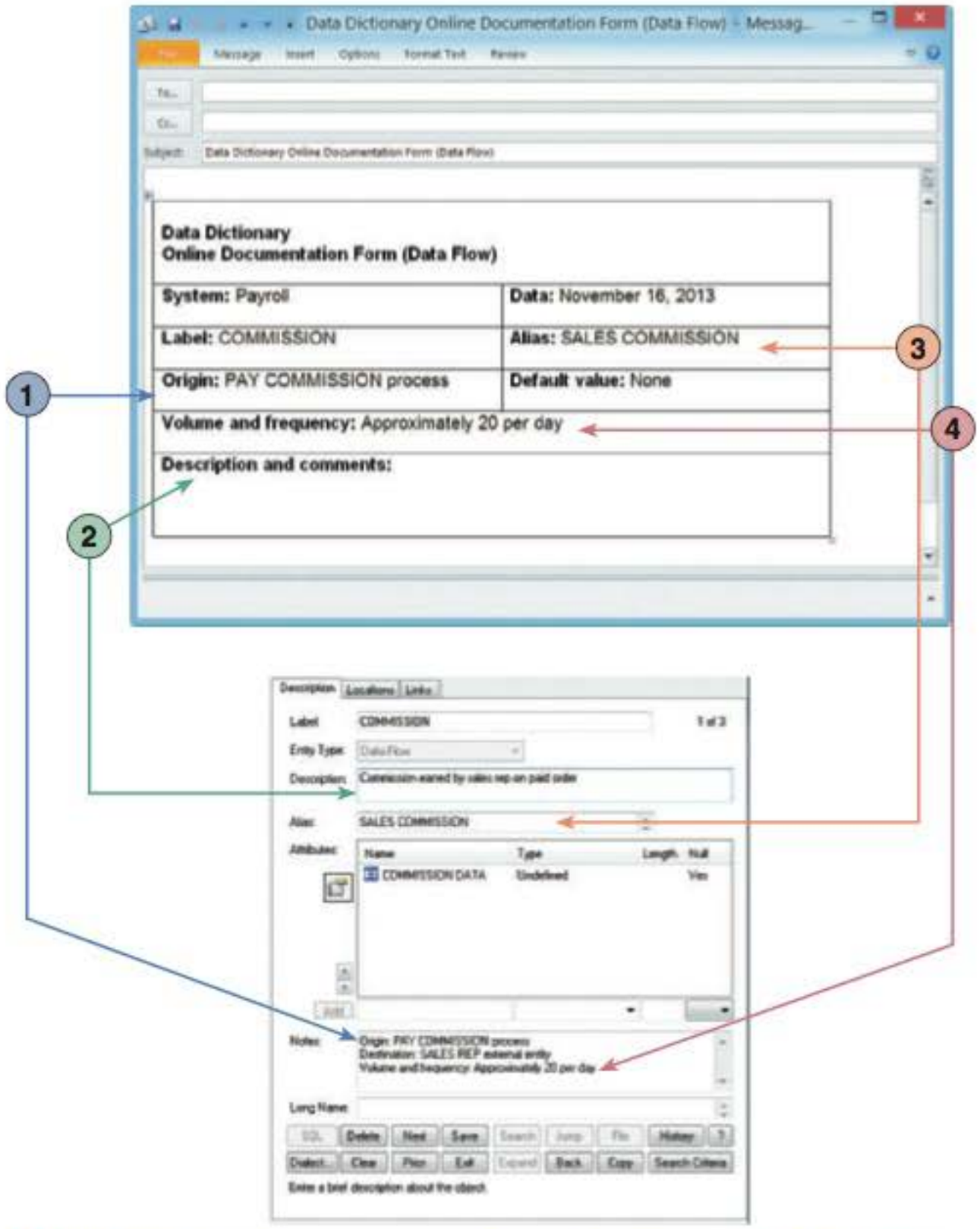


FIGURE 5-25 In the upper screen, an analyst has entered four items of information in an online documentation form. The lower screen shows the same four items entered into a Visible Analyst data dictionary form.

Screenshots used with permission from Visible Systems Corporation.

Documenting the Data Stores

You must document every DFD data store in the data dictionary. Figure 5-26 on the next page shows the definition of a data store named IN STOCK.

1. This data store has an alternative name, or alias.
2. For consistency, data flow names are standardized throughout the data dictionary.
3. It is important to document these estimates because they will affect design decisions in subsequent SDLC phases.

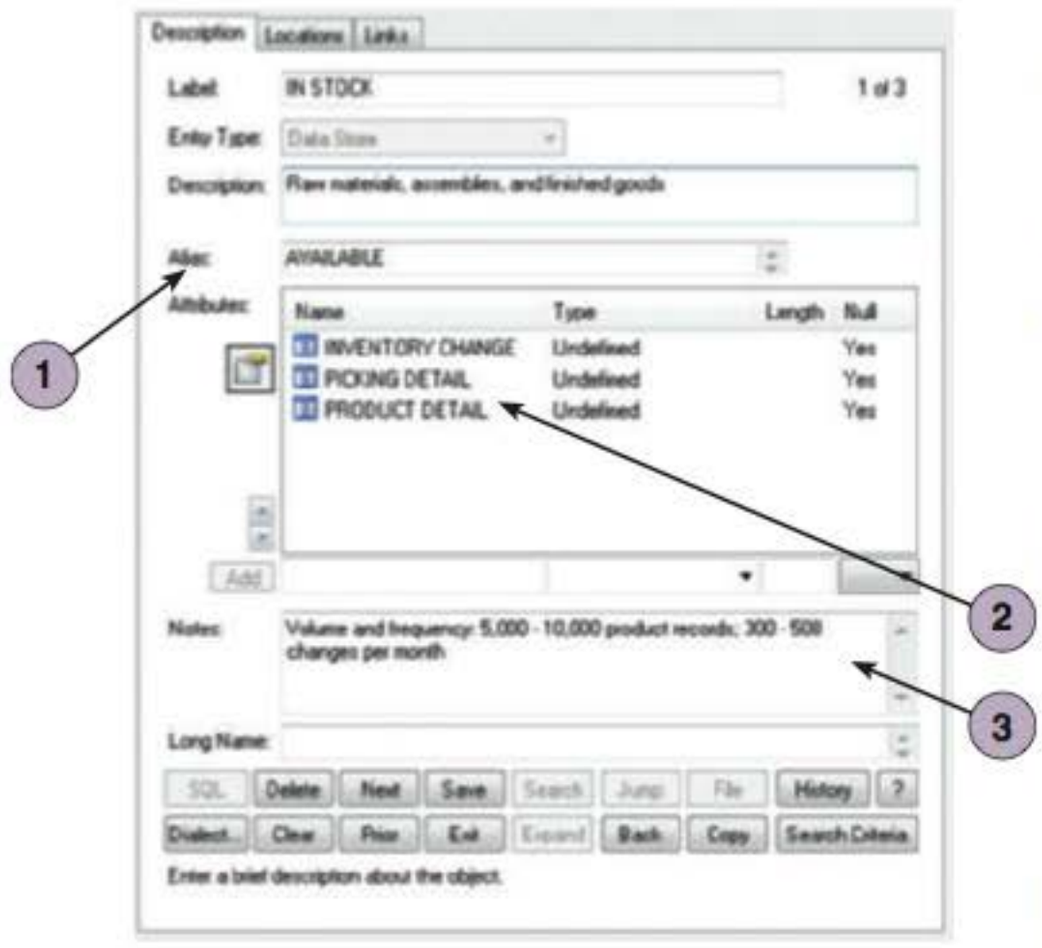


FIGURE 5-26 Visible Analyst screen that documents a data store named IN STOCK.

Screenshot used with permission from Visible Systems Corporation.

Typical characteristics of a data store are as follows:

Data store name or label. The data store name as it appears on the DFDs.

Description. Describes the data store and its purpose.

Alternate name(s). Aliases for the DFD data store name.

Attributes. Standard DFD names that enter or leave the data store.

Volume and frequency. Describes the estimated number of records in the data store and how frequently they are updated.

Documenting the Processes

You must document every process, as shown in Figure 5-27. Your documentation includes a description of the process's characteristics and, for functional primitives, a process description, which is a model that documents the processing steps and business logic.

The following are typical characteristics of a process:

Process name or label. The process name as it appears on the DFDs.

Description. A brief statement of the process's purpose.

Process number. A reference number that identifies the process and indicates relationships among various levels in the system.

Process description. This section includes the input and output data flows. For functional primitives, the process description also documents the processing steps and business logic. You will learn how to write process descriptions in the next section.

1. The process number identifies this process. Any subprocesses are numbered 1.1, 1.2, 1.3, and so on.
2. These data flows will be described specifically elsewhere in the data dictionary.

FIGURE 5-27 Visible Analyst screen that describes a process named VERIFY ORDER.

Screenshot used with permission from Visible Systems Corporation.

Documenting the Entities

By documenting all entities, the data dictionary can describe all external entities that interact with the system. Figure 5-28 shows a definition for an external entity named WAREHOUSE.

Typical characteristics of an entity include the following:

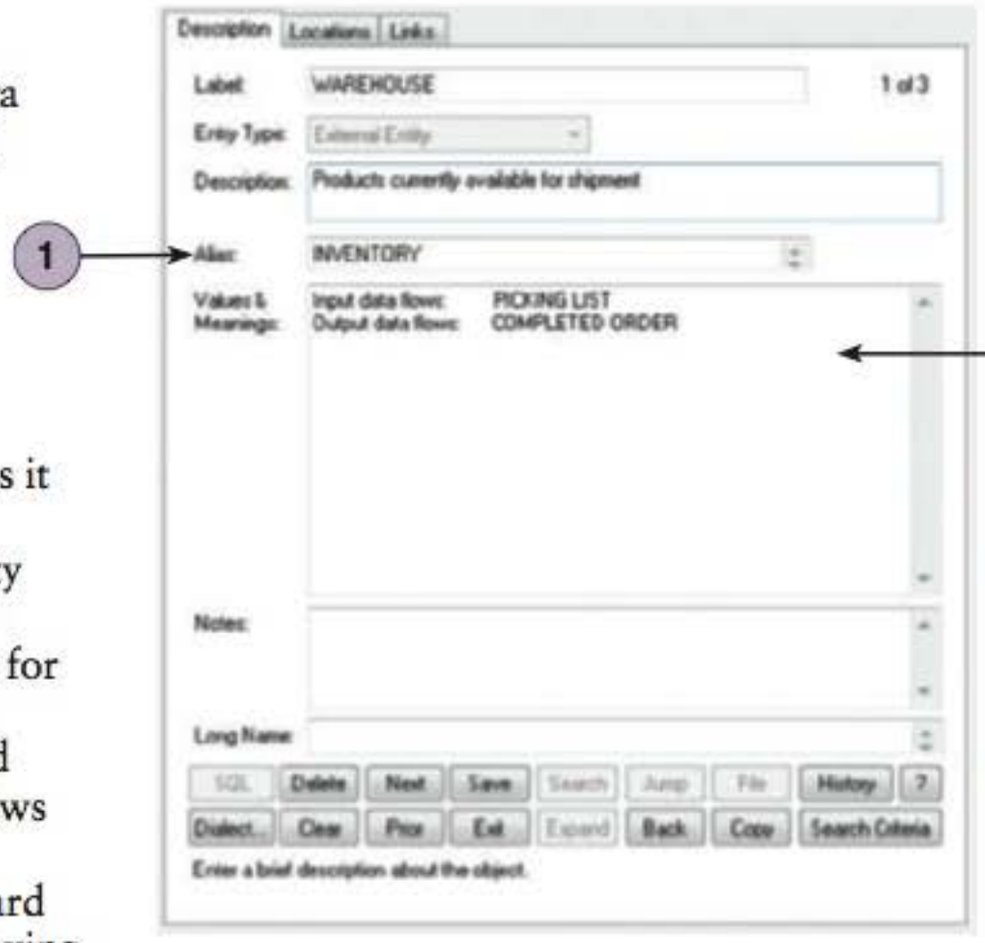
Entity name. The entity name as it appears on the DFDs.

Description. Describe the entity and its purpose.

Alternate name(s). Any aliases for the entity name.

Input data flows. The standard DFD names for the input data flows to the entity.

Output data flows. The standard DFD names for the data flows leaving the entity.



1. The external entity also can have an alternative name, or alias, if properly documented.
2. For consistency, these data flow names are standardized throughout the data dictionary.

FIGURE 5-28 Visible Analyst screen that documents an external entity named WAREHOUSE.

Screenshot used with permission from Visible Systems Corporation.

Documenting the Records

A record is a data structure that contains a set of related data elements that are stored and processed together. Data flows and data stores consist of records that you must document in the data dictionary. You define characteristics of each record, as shown in Figure 5-29.

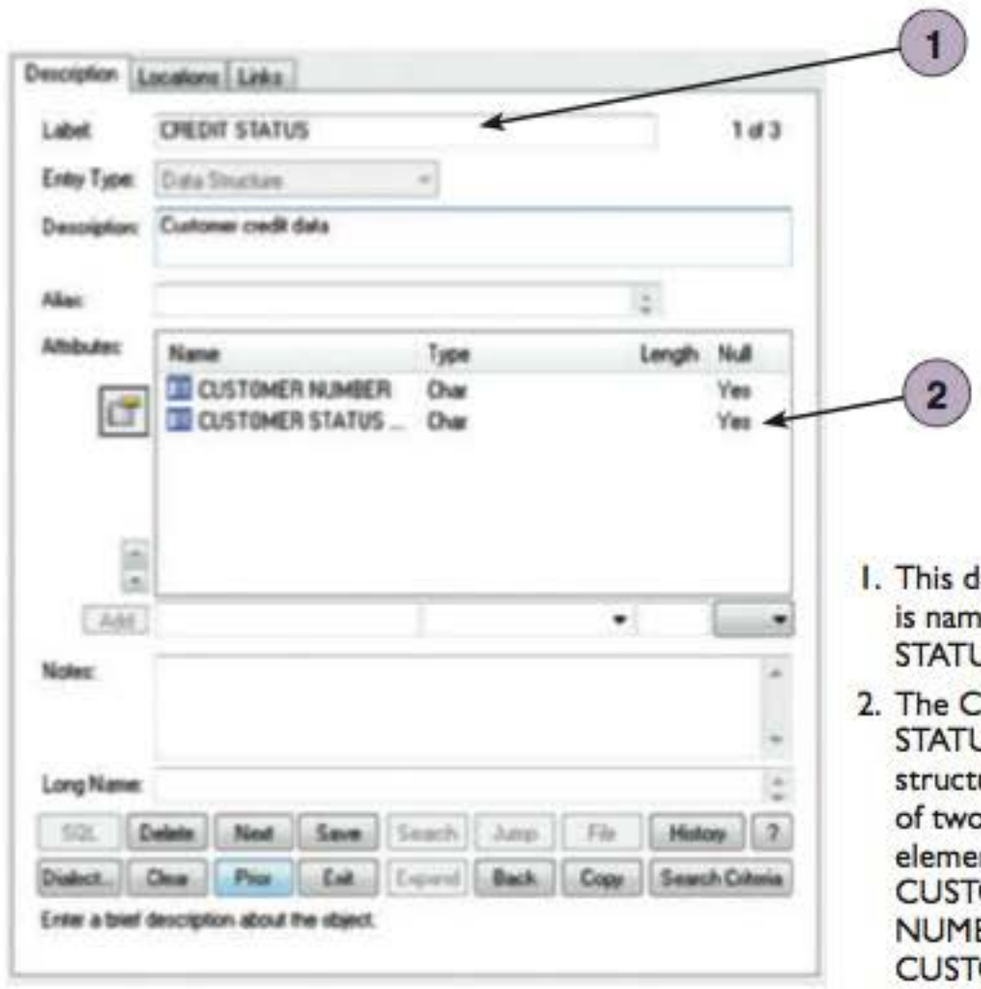
Typical characteristics of a record include the following:

Record or data structure name. The record name as it appears in the related data flow and data store entries in the data dictionary.

Definition or description. A brief definition of the record.

Alternate name(s). Any aliases for the record name.

Attributes. A list of all the data elements included in the record. The data element names must match exactly what you entered in the data dictionary.



1. This data structure is named CREDIT STATUS.
2. The CREDIT STATUS data structure consists of two data elements: CUSTOMER NUMBER and CUSTOMER STATUS CODE.

FIGURE 5-29 Visible Analyst screen that documents a record, or data structure named CREDIT STATUS.

Screenshot used with permission from Visible Systems Corporation.

Data Dictionary Reports

The data dictionary serves as a central storehouse of documentation for an information system. A data dictionary is created when the system is developed, and is updated constantly as the system is implemented, operated, and maintained. In addition to describing each data element, data flow, data store, record, entity, and process, the data

dictionary documents the relationships among these components. You can obtain many valuable reports from a data dictionary, including the following:

- An alphabetized list of all data elements by name
- A report describing each data element and indicating the user or department that is responsible for data entry, updating, or deletion
- A report of all data flows and data stores that use a particular data element
- Detailed reports showing all characteristics of data elements, records, data flows, processes, or any other selected item stored in the data dictionary

Process Description Tools

A **process description** documents the details of a functional primitive, and represents a specific set of processing steps and business logic. Using a set of process description tools, you create a model that is accurate, complete, and concise. Typical process description tools include structured English, decision tables, and decision trees. When you analyze a functional primitive, you break the processing steps down into smaller units in a process called modular design.

It should be noted that this chapter deals with structured analysis, but the process description tools also can be used in object-oriented development, which is described in Chapter 6. You learned in Chapter 1 that O-O analysis combines data and the processes that act on the data into things called objects, that similar objects can be grouped together into classes, and that O-O processes are called methods. Although O-O programmers use different terminology, they create the same kind of modular coding structures, except that the processes, or methods, are stored inside the objects, rather than as separate components.

FIGURE 5-30 Sequence structure.

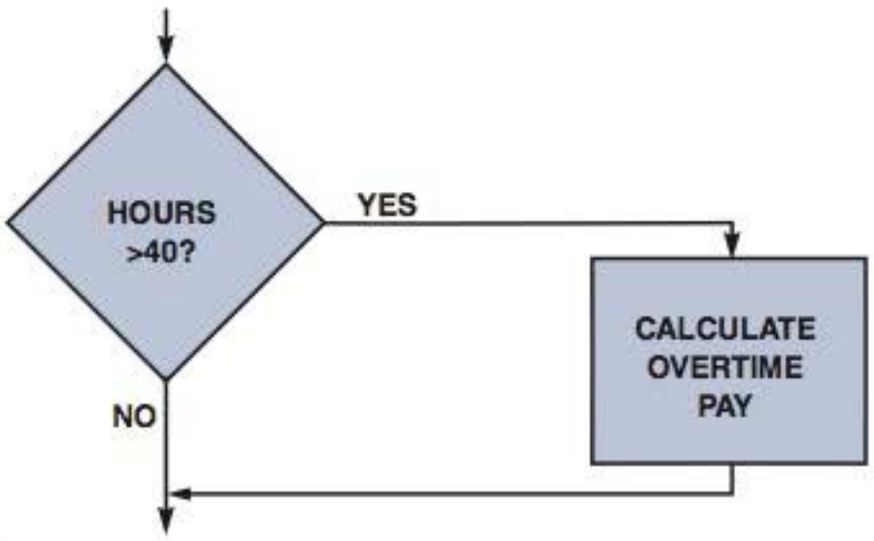


FIGURE 5-31 Selection structure.

Modular Design

Modular design is based on combinations of three **logical structures**, sometimes called **control structures**, which serve as building blocks for the process. Each logical structure must have a single entry and exit point. The three structures are called sequence, selection, and iteration. A rectangle represents a step or process, a diamond shape represents a condition or decision, and the logic follows the lines in the direction indicated by the arrows.

1. **Sequence.** The completion of steps in sequential order, one after another, as shown in Figure 5-30. One or more of the steps might represent a subprocess that contains additional logical structures.
2. **Selection.** The completion of one of two or more process steps based on the results of a test or condition. In the example shown in Figure 5-31, the system tests the input, and if the hours are greater than 40, it performs the CALCULATE OVERTIME PAY process.

3. **Iteration.** The completion of a process step that is repeated until a specific condition changes, as shown in Figure 5-32. An example of iteration is a process that continues to print paychecks until it reaches the end of the payroll file. Iteration also is called **looping.**

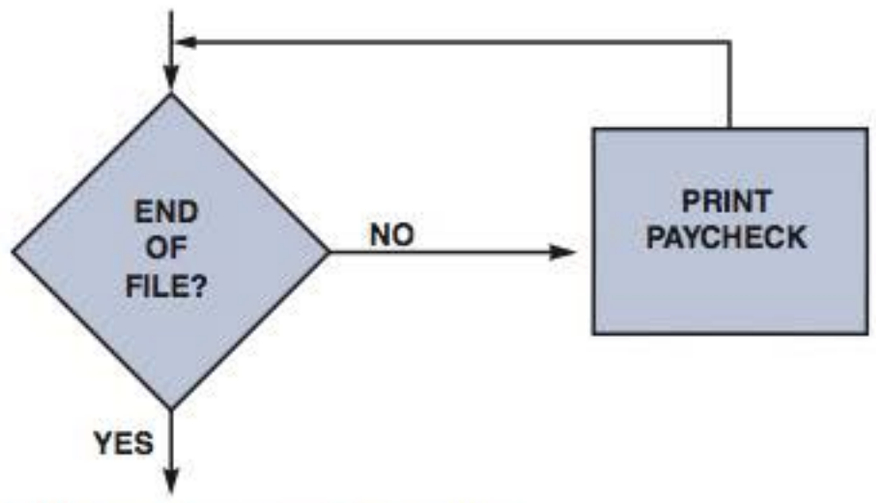


FIGURE 5-32 Iteration structure.

Sequence, selection, and iteration structures can be combined in various ways to describe processing logic.

Structured English

Structured English is a subset of standard English that describes logical processes clearly and accurately. When you use structured English, you must conform to the following rules:

- Use only the three building blocks of sequence, selection, and iteration.
- Use indentation for readability.
- Use a limited vocabulary, including standard terms used in the data dictionary and specific words that describe the processing rules.

An example of structured English appears in Figure 5-33, which shows the VERIFY ORDER process that was illustrated earlier. Notice that the structured English version documents the actual logic that will be coded into the system. Structured English can help make your process descriptions accurate and understandable to users and system developers.

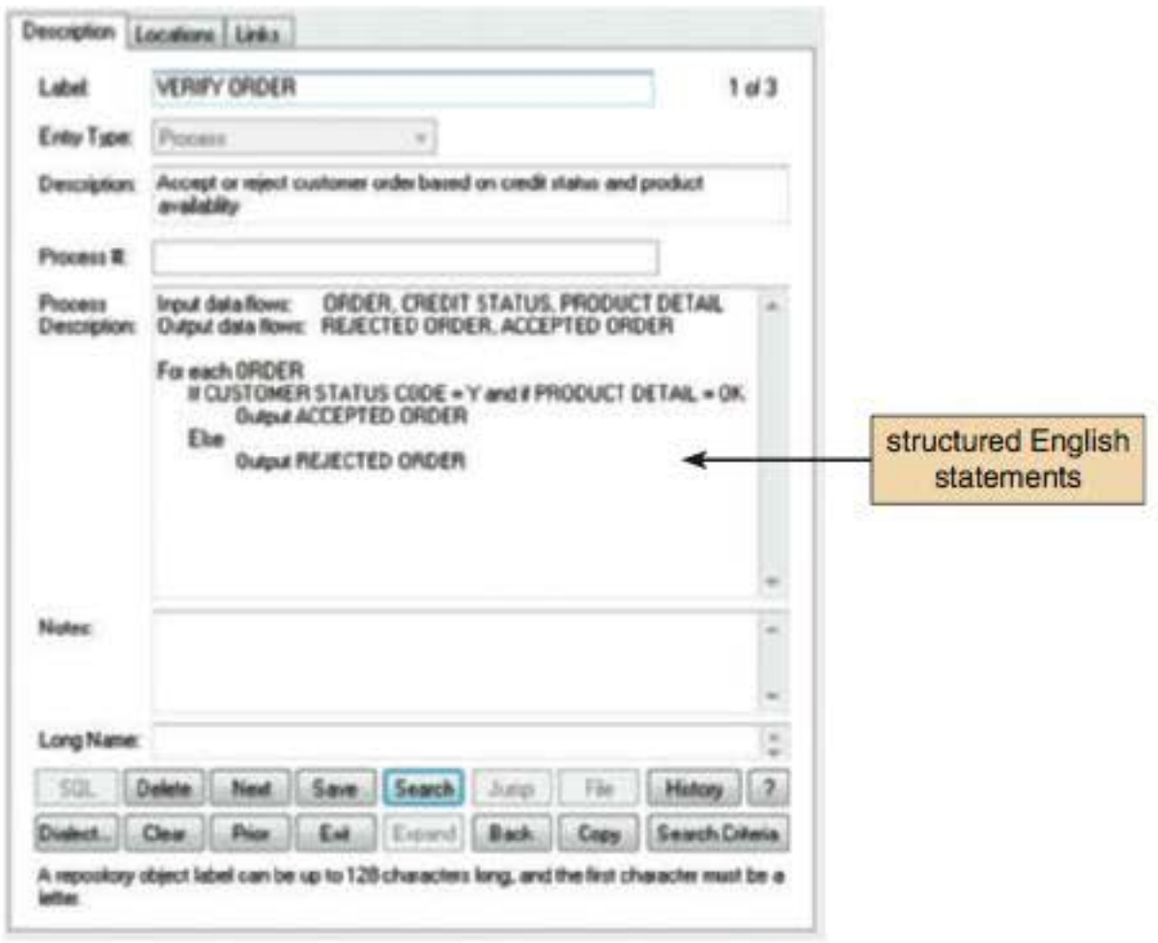


FIGURE 5-33 The VERIFY ORDER process description includes logical rules and a structured English version of the policy. Notice the alignment and indentation of the logic statements.

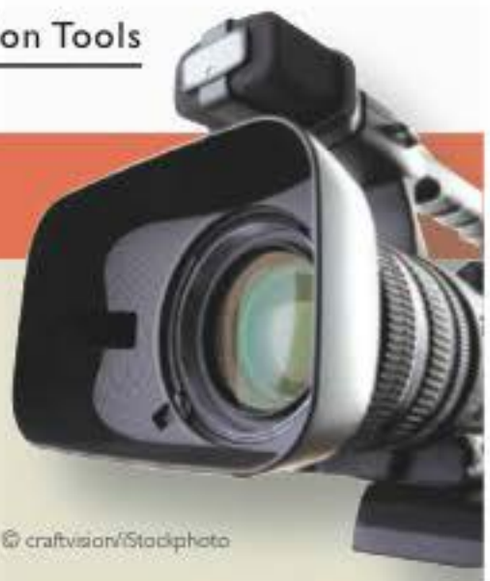
Structured English might look familiar to programming students because it resembles **pseudocode,** which is used in program design. Although the techniques are similar, the primary purpose of structured English is to describe the underlying business logic, while programmers, who are concerned with coding, mainly use pseudocode as a shorthand notation for the actual code.

Figure 5-34 shows another example of structured English. After you study the sales promotion policy, notice that the structured English version describes the processing logic that the system must apply. Following structured English rules ensures that your process descriptions are understandable to users who must confirm that the process is correct, as well as to other analysts and programmers who must design the information system from your descriptions.

Video Learning Session
Decision Tables

If you have an MIS CourseMate access code, you can launch interactive Video Learning Sessions to help you understand systems development concepts and practice your skills. You can watch the sessions on your computer or mobile device, and pause, rewind, or replay a video at any time. To log on to the MIS CourseMate site at **www.cengagebrain.com**, you must create a student account and then register this book.

This session is about decision tables, why they are important process description tools, how to create decision tables, and how to analyze conditions and outcomes in a decision table.



© craftvision/iStockphoto

Decision Tables

A **decision table** is a logical structure that shows every combination of conditions and outcomes. Analysts often use decision tables to describe a process and ensure that they have considered all possible situations. You can create decision tables using Microsoft PowerPoint, Word, or Excel.

TABLES WITH ONE CONDITION If a process has a single condition, there only are two possibilities – *yes* or *no*. Either the condition is present or it is not, so there are only two rules. For example, to trigger an overtime calculation, the process condition might be: *Are the hours greater than 40?* If so, the calculation is made. Otherwise, it is not.

TABLES WITH TWO CONDITIONS Suppose you want to create a decision table based on the Verify Order business process shown in Figure 5-34. When documenting a process, it is important to ensure that you list every possibility. In this example, the process description contains two conditions: product stock status and customer credit status. If *both* conditions are met, the order is accepted. Otherwise the order is rejected.

After you identify all the conditions and outcomes, you are ready to create a decision table similar to the one shown in Figure 5-35. To create the table, follow the four steps listed in the margin.

1. Place the name of the process in a heading at the top left.
2. Enter the conditions under the heading, with one condition per line, to represent the customer status and availability of products.
3. Enter all potential combinations of Y/N (for yes and no) for the conditions. Each column represents a numbered possibility called a rule.
4. Place an X in the action entries area for each rule to indicate whether to accept or reject the order.

VERIFY ORDER Business Process with Two Conditions
• An order will be accepted only if the product is in stock and the customer's credit status is OK.
• All other orders will be rejected.

FIGURE 5-34 The Verify Order business process has two conditions. For an order to be accepted, the product must be in stock and the customer must have an acceptable credit status.

VERIFY ORDER Process

	1	2	3	4	
Credit status is OK	Y	Y	N	N	3
Product is in stock	Y	N	Y	N	
Accept order	X				4
Reject order		X	X	X	

FIGURE 5-35 Example of a simple decision table showing the processing logic of the VERIFY ORDER process.

Notice that each condition has two possible values, so the number of rules doubles each time you add another condition. For example, one condition creates two rules, two conditions create four rules, three conditions create eight rules, and so on. In the two-condition example in Figure 5-35, four possibilities exist, but Rule 1 is the *only* combination that will accept an order.

TABLES WITH THREE CONDITIONS Suppose the company now decides that the credit manager can waive the customer credit requirement, as shown in Figure 5-36. That creates a third condition, so there will be eight possible rules. The new decision table might resemble the one shown in Figure 5-37.

VERIFY ORDER Business Process with Three Conditions

- An order will be accepted only if the product is in stock and the customer's credit status is OK.
- The credit manager can waive the credit status requirement.
- All other orders will be rejected.

FIGURE 5-36 A third condition has been added to the Verify Order business process. For an order to be accepted, the product must be in stock and the customer must have an acceptable credit status. However, the credit manager now has the authority to waive the credit status requirement.

VERIFY ORDER Process with Credit Waiver (initial version)

	1	2	3	4	5	6	7	8
Credit status is OK	Y	Y	Y	Y	N	N	N	N
Product is in stock	Y	Y	N	N	Y	Y	N	N
Waiver from credit manager	Y	N	Y	N	Y	N	Y	N
Accept order	X	X			X			
Reject order			X	X		X	X	X

FIGURE 5-37 This table is based on the Verify Order conditions shown in Figure 5-36. With three conditions, there are eight possible combinations, or rules.

First, you must fill in the Y-N patterns, as shown in Figure 5-37. The best way to assure that all combinations appear is to use patterns like these. The first condition uses a pattern of Y-Y-Y-Y followed by N-N-N-N; the second condition uses a repeating Y-Y-N-N pattern; and the pattern in the third condition is a series of Y-Ns.

The next step is very important, regardless of the number of conditions. Each numbered column, or rule, represents a different set of conditions. You must analyze the logic carefully and show the outcome for each rule. Before going further, study the table in Figure 5-37 and be sure you understand the logical outcome for each of the eight rules.

When all the outcomes have been determined, you are ready to simplify the table. In a multicondition table, some rules might be duplicates, redundant, or unrealistic. To simplify the table, follow these steps:

1. Study each combination of conditions and outcomes. When you have rules with three conditions, only one or two of them may control the outcome, and the other conditions simply do not matter.
2. If you identify conditions that do not affect the outcome, mark them with dashes (-) as shown in the first table in Figure 5-38 on the next page.
3. Now combine and renumber the rules, as shown in the second table in Figure 5-38.

VERIFY ORDER Process with Credit Waiver (with rules marked for combination)

	1	2	3	4	5	6	7	8
Credit status is OK	Y	Y	-	-	N	N	-	-
Product is in stock	Y	Y	N	N	Y	Y	N	N
Waiver from credit manager	-	-	-	-	Y	N	-	-
Accept order	X	X			X			
Reject order			X	X		X	X	X

1. Because the product is not in stock, the other conditions do not matter.
2. Because the other conditions are met, the waiver does not matter.

VERIFY ORDER Process with Credit Waiver (after rule combination and simplification)

	1 (COMBINES PREVIOUS 1, 2)	2 (PREVIOUS 5)	3 (PREVIOUS 6)	4 (COMBINES PREVIOUS 3, 4, 7, 8)
Credit status is OK	Y	N	N	-
Product is in stock	Y	Y	Y	N
Waiver from credit manager	-	Y	N	-
Accept order	X	X		
Reject order			X	X

FIGURE 5-38 In the first table, dashes have been added to indicate that a condition is not relevant. In the second version, rules have been combined. Notice that in final version, only four rules remain. These rules document the logic, and will be transformed into program code when the system is developed.

If you follow these steps, you will see that Rules 1 and 2 can be combined because credit status is OK in both rules, so the waiver would not matter. Rules 3, 4, 7, and 8 also can be combined because the product is not in stock, so other conditions do not matter. The result is that instead of eight possibilities, only four logical rules control the Verify Order process.

MULTIPLE OUTCOMES In addition to multiple conditions, decision tables can have more than two possible outcomes. For example, the sales promotion policy shown in Figure 5-39 includes three conditions: Was the customer a preferred customer, did the customer order $1,000 or more, and did the customer use our company charge card? Based on these conditions, four possible actions can occur, as shown in the decision table in Figure 5-40.

SALES PROMOTION POLICY – Holiday Season, 2014
• Preferred customers who order $1,000 or more are entitled to a 5% discount, and an additional 5% discount if they use our charge card.
• Preferred customers who do not order $1,000 or more will receive a $25 bonus coupon.
• All other customers will receive a $5 bonus coupon.

FIGURE 5-39 A sales promotion policy with three conditions. Notice that the first statement contains two *separate* conditions – one for the 5% discount, and another for the additional discount.

Sales Promotion Policy (initial version)

	1	2	3	4	5	6	7	8
Preferred customer	Y	Y	Y	Y	N	N	N	N
Ordered $1,000 or more	Y	Y	N	N	Y	Y	N	N
Used our charge card	Y	N	Y	N	Y	N	Y	N
5% discount	X	X						
Additional 5% discount	X							
$25 bonus coupon			X	X				
$5 bonus coupon					X	X	X	X

FIGURE 5-40 This decision table is based on the sales promotion policy in Figure 5-39. This is the initial version of the table, before simplification.

As explained in the preceding section, most tables can be simplified, and this one is no exception. When you study the conditions and outcomes, you realize that:

- In Rule 1, all three conditions are met, so *both* 5% discounts apply.
- In Rule 2, a preferred customer orders $1,000 or more, but does not use our charge card, so only *one* 5% discount applies.
- Rules 3 and 4 can be combined into a single rule. Why? If preferred customers do not order $1,000 or more, it does not matter whether they use our charge card – either way, they only earn a $25 bonus coupon. Therefore, Rules 3 and 4 really are a single rule.
- Rules 5, 6, 7, and 8 also can be combined into a single rule – because if the person is *not* a preferred customer, he or she can *only* receive a $5 bonus coupon, and the other conditions simply do not matter. So, you insert a dash if a condition is irrelevant, as shown in Figure 5-41.

If you add dashes for rules that are not relevant, your table should resemble the one shown in Figure 5-41. When you combine and simplify the results, only four rules remain: Rule 1, Rule 2, Rule 3 (a combination of initial Rules 3 and 4), and Rule 4 (a combination of initial Rules 5, 6, 7, and 8).

Sales Promotion Policy (final version)

	1	2	3	4	5	6	7	8
Preferred customer	Y	Y	Y	Y	N	N	N	N
Ordered $1,000 or more	Y	Y	N	N	-	-	-	-
Used our charge card	Y	N	-	-	-	-	-	-
5% discount	X	X						
Additional 5% discount	X							
$25 bonus coupon			X	X				
$5 bonus coupon					X	X	X	X

FIGURE 5-41 In this version, dashes have been added to indicate that a condition is not relevant. At this point, it appears that several rules can be combined.

Decision tables often are the best way to describe a complex set of conditions. Many analysts use decision tables because they are easy to construct and understand, and programmers find it easy to work from a decision table when developing code.

CASE IN POINT 5.2: ROCK SOLID OUTFITTERS (PART 1)

Leah Jones is the IT manager at Rock Solid Outfitters, a medium-sized supplier of outdoor climbing and camping gear. Steve Allen, the marketing director, has asked Leah to develop a special Web-based promotion. As Steve described it to Leah, Rock Solid will provide free shipping for any customer who either completes an online survey form or signs up for the Rock Solid online newsletter. Additionally, if a customer completes the survey *and* signs up for the newsletter, Rock Solid will provide a $10 merchandise credit for orders of $100 or more. Leah has asked you to develop a decision table that will reflect the promotional rules that a programmer will use. She wants you to show all possibilities, and then to simplify the results to eliminate any combinations that would be unrealistic or redundant.

Decision Trees

A **decision tree** is a graphical representation of the conditions, actions, and rules found in a decision table. Decision trees show the logic structure in a horizontal form that resembles a tree with the roots at the left and the branches to the right. Like flowcharts, decision trees are useful ways to present the system to management. Decision trees and decision tables provide the same results, but in different forms. In many situations, a graphic is the most effective means of communication.

Figure 5-42 is based on the sales promotion policy shown in Figure 5-39 on page 208. A decision tree is read from left to right, with the conditions along the various branches and the actions at the far right. Because the example has two conditions with four resulting sets of actions, the example has four terminating branches at the right side of the tree.

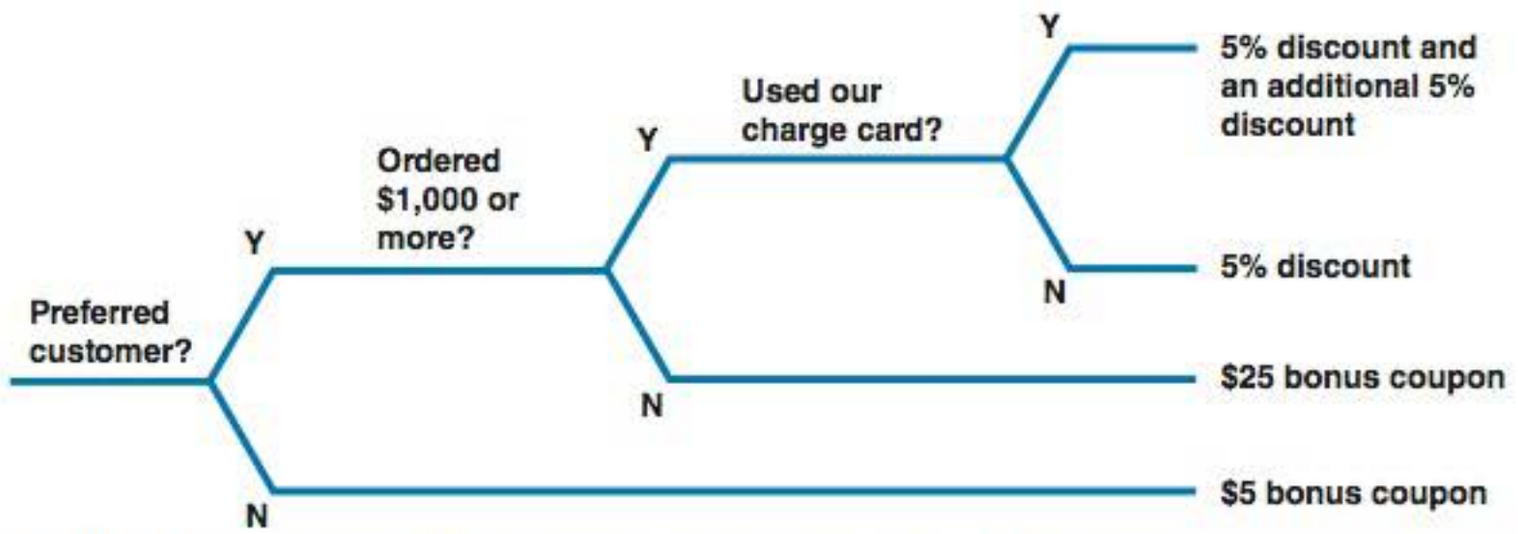


FIGURE 5-42 This example is based on the same Sales Promotion Policy shown in the decision tables in Figures 5-40 and 5-41 on the previous page. Like a decision table, a decision tree shows all combinations of conditions and outcomes. The main difference is the graphical format, which many viewers find easier to interpret.

Whether to use a decision table or a decision tree often is a matter of personal preference. A decision table might be a better way to handle complex combinations of conditions. On the other hand, a decision tree is an effective way to describe a relatively simple process.

CASE IN POINT 5.3: ROCK SOLID OUTFITTERS (PART 2)

Leah Jones, the IT manager at Rock Solid Outfitters, thinks you did a good job on the decision table task she assigned to you. Now she wants you to use the same data to develop a decision tree that will show all the possibilities for the Web-based promotion described in Part 1 of the case. She also wants you to discuss the pros and cons of decisions tables versus decision trees.

LOGICAL VERSUS PHYSICAL MODELS

While structured analysis tools are used to develop a logical model for a new information system, such tools also can be used to develop physical models of an information system. A physical model shows how the system's requirements are implemented. During the systems design phase, you create a physical model of the new information system that follows from the logical model and involves operational tasks and techniques.

Sequence of Models

What is the relationship between logical and physical models? Think back to the beginning of the systems analysis phase when you were trying to understand the existing system. Rather than starting with a logical model, you first studied the physical operations of the existing system to understand how the current tasks were carried out. Many systems analysts create a physical model of the current system and then develop a logical model of the current system before tackling a logical model of the new system. Performing that extra step allows them to understand the current system better.

Four-Model Approach

Many analysts follow a **four-model approach**, which means that they develop a physical model of the current system, a logical model of the current system, a logical model of the new system, and a physical model of the new system. The major benefit of the four-model approach is that it gives you a clear picture of current system functions before you make any modifications or improvements. That is important because mistakes made early in systems development will affect later SDLC phases and can result in unhappy users and additional costs. Taking additional steps to avoid these potentially costly mistakes can prove to be well worth the effort. Another advantage is that the requirements of a new information system often are quite similar to those of the current information system, especially where the proposal is based on new computer technology rather than a large number of new requirements. Adapting the current system logical model to the new system logical model in these cases is a straightforward process.

The only disadvantage of the four-model approach is the added time and cost needed to develop a logical and physical model of the current system. Most projects have very tight schedules that might not allow time to create the current system models. Additionally, users and managers want to see progress on the new system — they are much less concerned about documenting the current system. As a systems analyst, you must stress the importance of careful documentation and resist the pressure to hurry the development process at the risk of creating serious problems later.

CASE IN POINT 5.4: TIP TOP STAFFING

Tip Top Staffing supplies employees to hundreds of IT firms that require specialized skills for specific projects. Systems analysts Lisa Nuevo and Bill Goodman are working on the logical model of Tip Top's billing and records system, using DFDs, a data dictionary, and process descriptions. At some point while working on the logical model of the system, Lisa felt that some improvements should be made in the data forms that Tip Top uses to obtain information about job applicants. Was the subject of improving the forms a physical implementation issue? Is Lisa going off on a tangent by considering *how* something will be done, instead of sticking to *what* will be done?

A QUESTION OF ETHICS

© faberfoto_it/Stockphoto

This is your first week in your new job at Safety Zone, a leading producer of IT modeling software. Your prior experience with a smaller competitor gave you an edge in landing the job, and you are excited about joining a larger company in the same field.

So far, all is going well and you are getting used to the new routine. However, you are concerned about one issue. In your initial meeting with the IT manager, she seemed very interested in the details of your prior position, and some of her questions made you a little uncomfortable. She did not actually ask you to reveal any proprietary information, but she made it clear that Safety Zone likes to know as much as possible about its competitors.

Thinking about it some more, you try to draw a line between information that is OK to discuss, and topics such as software specifics or strategy that should be considered private. This is the first time you have ever been in a situation like this. How will you handle it?

CHAPTER SUMMARY

During data and process modeling, a systems analyst develops graphical models to show how the system transforms data into useful information. The end product of data and process modeling is a logical model that will support business operations and meet user needs. Data and process modeling involves three main tools: data flow diagrams, a data dictionary, and process descriptions.

Data flow diagrams (DFDs) graphically show the movement and transformation of data in the information system. DFDs use four symbols: The process symbol transforms data; the data flow symbol shows data movement; the data store symbol shows data at rest; and the external entity symbol represents someone or something connected to the information system. Various rules and techniques are used to name, number, arrange, and annotate the set of DFDs to make them consistent and understandable.

A set of DFDs is like a pyramid with the context diagram at the top. The context diagram represents the information system's scope and its external connections but not its internal workings. Diagram 0 displays the information system's major processes, data stores, and data flows and is the exploded version of the context diagram's process symbol, which represents the entire information system. Lower-level DFDs show additional detail of the information system through the leveling technique of numbering and partitioning. Leveling continues until you reach the functional primitive processes, which are not decomposed further and are documented with process descriptions. All diagrams must be balanced to ensure their consistency and accuracy.

The data dictionary is the central documentation tool for structured analysis. All data elements, data flows, data stores, processes, entities, and records are documented in the data dictionary. Consolidating documentation in one location allows you to verify the information system's accuracy and consistency more easily and generate a variety of useful reports.

Each functional primitive process is documented using structured English, decision tables, and decision trees. Structured English uses a subset of standard English that defines each process with combinations of the basic building blocks of sequence, selection, and iteration. You also can document the logic by using decision tables or decision trees.

Structured analysis tools can be used to develop a logical model during one systems analysis phase, and a physical model during the systems design phase. Many analysts use a four-model approach, which involves a physical model of the current system, a logical model of the current system, a logical model of the new system, and a physical model of the new system.

Key Terms

alias *199*
balancing *192*
black box *181*
black hole *183*
business logic *181*
business rules *181*
child diagram *191*
context diagram *188*
control structures *204*
data dictionary *197*
data element *197*
data flow *182*
data flow diagram (DFD) *180*
data item *197*
data repository *197*
data store *183*
data structures *197*
decision table *206*
decision tree *210*
decomposing *193*
diagram 0 *189*
diverging data flow *191*
domain *200*
entity *185*
exploding *193*
field *197*
four-model approach *211*
functional primitive *191*
Gane and Sarson *180*
gray hole *183*
iteration *205*
length *199*
leveling *192*
logical model *178*
logical structures *204*
looping *205*
modular design *204*
parent diagram *191*
partitioning *193*
physical model *178*
process *181*
process 0 *188*
process description *204*
pseudocode *205*
records *197*
selection *204*
sequence *204*
sink *185*
source *185*
spontaneous generation *183*
structured English *205*
terminators *185*
type *199*
validity rules *200*
Yourdon *180*

Chapter Exercises

Questions

1. Describe data and process modeling concepts and tools.
2. Explain the differences between Gane and Sarson and Yourdon symbols. Provide examples of symbols that represent processes, data flows, data stores, and entities.
3. What is the difference between a context diagram and diagram 0? Which symbol is *not* used in a context diagram?
4. How would you *explode* DFDs?
5. Describe a data dictionary and list the types of information it contains.
6. How would you level DFDs?
7. How would you balance DFDs?
8. What is the purpose of decision tables? How do you create them?
9. Why would a manager prefer a decision tree instead of a decision table?
10. What is structured English?

Discussion Topics

1. Suppose you were assigned to develop a logical model of the registration system at a school or college. Would you be better off using a top-down approach, or would a bottom-up strategy be better? What would influence your decision?
2. Some systems analysts find it better to start with a decision table, and then construct a decision tree. Others believe it is easier to do it in the reverse order. Which do you prefer? Why?
3. A systems analyst attended a weeklong workshop on structured analysis. When she returned to her job, she told her boss that structured analysis was not worth the time to learn and use on the job. Her view was that it was too academic and had too much new terminology to be useful. Do you agree? Why or why not?
4. This chapter describes a *black box* concept that allows more detail to be shown as a process is exploded. Can the concept be applied in business management generally, or is it limited to information systems design? Provide some reasons and examples.

Projects

1. Draw a context diagram and a diagram 0 DFD that represents the information system at a typical library.
2. On the Internet, locate at least three firms that offer CASE tools as shareware or free, public domain software. Describe what you found.
3. Create a decision table with three conditions. You can make one up, or use a scenario from everyday life. Either way, be sure to show all possible outcomes.
4. The data flow symbols shown on page 181 were designed by Ed Yourdon, a well-known IT author, lecturer, and consultant. Many IT professionals consider him to be among the most influential men and women in the software field. Learn more about Mr. Yourdon by visiting his Web site at **www.yourdon.com**, and write a brief review of his work.

Apply Your Knowledge

This section contains four mini-cases. Each case describes a situation, explains your role, and requires you to apply what you learned in the chapter.

Globe Consulting

You are a senior systems analyst at Globe Consulting, a growing IT consulting firm. You are leading the development team for a major client. You need to explain data and process modeling to your two newly hired junior analysts (Michelle and Aidan) before meeting with the client tomorrow afternoon.

Tasks

1. Describe the rules for creating DFDs.
2. What are the data flow and process combinations that must be avoided when creating a DFD?
3. Explain what a diagram 0 is and how it is used.
4. Ask Michelle and Aidan to review the order system context diagram on page 188, and compare it with the order system diagram 0 DFD on page 192. Then ask them to answer the following questions: (a) How many external entities are shown in each diagram? (b) In each diagram, how many data flows connect to the external entities? (c) How many processes are identified in the diagram 0 DFD? (d) Could the data store have been shown in the context diagram? Why or why not?

2 Kitchen Gadgets

Kitchen Gadgets sells a line of high-quality kitchen utensils and gadgets. When customers place orders on the company's Web site or through electronic data interchange (EDI), the system checks to see if the items are in stock, issues a status message to the customer, and generates a shipping order to the warehouse, which fills the order. When the order is shipped, the customer is billed. The system also produces various reports.

Tasks

1. List four elements used in DFDs, draw the symbols, and explain how they are used.
2. Draw a context diagram for the order system.
3. Draw a diagram 0 DFD for the order system.
4. Explain the importance of leveling and balancing.

3 Big State University

The Big State University course catalog reads as follows: "To enroll in MIS 260, which is an advanced course, a student must complete two prerequisites: MIS 120 and MIS 222. A student who completes either one of these prerequisites and obtains the instructor's permission, however, will be allowed to take MIS 260."

Tasks

1. Create a decision table that describes the Big State University course catalog regarding eligibility for MIS 260. Show all possible rules.
2. Simplify the table you just created. Describe the results.
3. Draw a simplified decision tree to represent the Big State University catalog. Describe the results.
4. Why might you use a decision tree rather than a decision table?

4 Fresh-Mart Grocery

Fresh-Mart Grocery is a regional grocery chain that is developing an information system to monitor inventory levels, product sales, and merchandise turnover. As products are sold in stores, they need to be replaced by shipments from the distribution centers. The IT manager wants you to document a process called BALANCE that determines whether extra product inventory is needed in a particular store. The BALANCE process is to be used in a just-in-time inventory system that automatically orders products and routes them to various stores for sale. A store manager can override the automatic BALANCE process if he or she so desires to get more product into a store.

Tasks

1. Create a decision table that describes the movement of inventory.
2. Draw a decision tree that describes the merchandise inventory management process.
3. Name four attributes that you can use to define a data flow in the grocery inventory information system.
4. Name four attributes that you can use to define a data store in the grocery inventory information system.

Case Studies

Each chapter includes a Chapter Case, a Continuing Case, a Capstone Case, and an Online Case Simulation. You can learn more about the Online Case Simulation in the MIS CourseMate Features section.

Chapter Case: College Driver Insurance

College Driver Insurance offers discounted auto insurance to college students.

Background

The company uses several factors to set discounts, as follows:

- A student who has taken a safe driver course earns a 5% discount.
- A student with a clean record (no tickets or accidents in the last three years) earns a 5% discount.
- A student who is 23 or older earns a 5% discount.
- A student who meets all three conditions (has taken a safe driver course *and* has a clean record *and* is 23) earns an additional bonus discount of 5%.

Tasks

1. Create a decision table that describes the discount rules.
2. Simplify the table you just created. Describe the results.
3. Draw a simplified decision tree that shows the discount rules.
4. What are the pros and cons of using a decision table versus a decision tree?

Continuing Case: Personal Trainer, Inc.

Personal Trainer, Inc. owns and operates fitness centers in a dozen Midwestern cities. The centers have done well, and the company is planning an international expansion by opening a new "supercenter" in the Toronto area. Personal Trainer's president, Cassia Umi, hired an IT consultant, Susan Park, to help develop an information system for the new facility. During the project, Susan will work closely with Gray Lewis, who will manage the new operation.

Background

Susan Park has completed a preliminary investigation and performed the fact-finding tasks that were described in Chapters 2 and 4. Now, she will use the results to develop a logical model of the proposed information system.

Tasks

Before you perform the following tasks, you should review the information provided in Chapters 2 and 4 of the case.

1. Prepare a context diagram for the new system.
2. Prepare a diagram 0 DFD for the new system.
3. Write a brief memo that explains at least five attributes that you can use to define a process in the order system.
4. Write a brief memo that explains at least five attributes that you can use to define an entity in the order system.

Capstone Case: New Century Wellness Group

New Century Wellness Group offers a holistic approach to healthcare with an emphasis on preventive medicine as well as traditional medical care. In your role as an IT consultant, you will help New Century develop a new information system.

Background

You began the systems analysis phase by conducting interviews, reviewing existing reports, and observing office operations. (Your instructor may provide you with a sample set of interview summaries.)

The New Century medical team performs services and medical procedures, which are coded according to the American Medical Association's Current Procedure Terminology (CPT). CPT codes consist of five numeric digits and a two-digit suffix, and most insurance payers require the codes to be included with billing information.

The new system must be able to handle the new ICD-10 procedure coding system, which will be required by the Centers for Medicare & Medicaid Services (CMS) beginning October 1, 2014. ICD-10 codes consist of seven alphanumeric characters, which can be electronically transmitted and received. New Century's information system must interface with 25 California health insurance providers. The new system represents an opportunity for significant cost saving for New Century, and more convenience for patients, who will be able to go online to update medical information, schedule appointments, and request medical records.

During your fact-finding, you learned that the clinic requires various reports, as follows:

- Daily appointment list for each provider. The list shows all scheduled appointment times, patient names, and services to be performed, including the procedure code and description.
- Daily report call list, which shows the patients who are to be reminded of their next day's appointments. The call list includes the patient name, telephone number, appointment time, and provider name.
- Weekly provider report that lists each of the providers and the weekly charges generated, plus a month-to-date (MTD) and a year-to-date (YTD) summary as well as profit distribution data for the partners.
- Monthly patient statement, which includes the statement date, head of household name and address, previous month's balance, total household charges MTD, total payments MTD, and the current balance. The bottom section of the statement shows activity for the month in date order. For each service performed, a line shows the patient's name, the service date, the procedure code and description, and the charge. The statement also shows the date and amount of all payments and insurance claims. When an insurance payment is received, the source and amount are noted on the form. If the claim is denied or only partially paid, a code is used to explain the reason. A running balance appears at the far right of each activity line.
- Weekly Insurance Company Report.
- Monthly Claim Status Summary.

In addition to these reports, the office staff would like automated e-mail and text messaging capability for sending reminders to patients when it is time to schedule an appointment. Data also needs to be maintained on employers who participate in employee wellness programs. This information can be used for marketing purposes throughout the year. Finally, the new system needs to track employee schedules, attendance, vacation time, and paid time off.

Now you are ready to organize the facts and prepare a system requirements document that represents a logical model of the proposed system. Your tools will include DFDs, a data dictionary, and process descriptions.

Capstone Case: New Century Wellness Group

Tasks

1. Prepare a context diagram for New Century's information system.
2. Prepare a diagram 0 DFD for New Century. Be sure to show numbered processes for handling appointment processing, payment and insurance processing, report processing, and records maintenance. Also, prepare lower-level DFDs for each numbered process.
3. Prepare a list of data stores and data flows needed for the system. Under each data store, list the data elements required.
4. Prepare a data dictionary entry and process description for one of the system's functional primitives.

CASE Tool Workshop

Systems analysts use CASE tools to help them plan, build, and maintain information systems. To learn more about CASE tools, turn to Part B of the Toolkit that follows Chapter 12. You can complete these tasks with the Visible Analyst® CASE tool, which is available with this textbook, or a similar tool.

Background

You decided to start a mobile auto detailing service that will visit customers at their homes or businesses, and use traditional, high-quality hand washing and waxing methods. In addition to retail customers, several auto dealers have expressed interest in your services, and you are eager to get started. You plan to hire college students as technicians, train and supply them, and offer them a share of the profits as an incentive.

Your friend, who is an IT major, offered to help you set up an information system by customizing several Microsoft Office applications to meet your needs. She suggested that you prepare an initial context diagram and a DFD diagram 0, and not to be too concerned about the details — just the main entities, data flows, and data stores. Thinking it over, you know that you will have employees, retail customers, auto dealer accounts, one or more suppliers, and your local bank, where you plan to set up a business checking account to handle the business. You also plan to keep track of customers, employee information, profit-sharing data, and an operations log, among other things.

Tasks

1. Create a context diagram from the new business. It is okay to use your imagination with respect to data flows that you envision.
2. Create a diagram 0 that includes at least three processes and two data stores that would be typical in a small business of this kind. You can use your imagination with respect to the processes and data stores that might be needed.

MIS CourseMate Features

If you have an MIS CourseMate access code, you can reinforce and extend your learning with premium content created for this textbook. For example, you can launch interactive Video Learning Sessions to help you understand systems development concepts and practice your skills. In addition, you can use MindTap Reader, which is a full, interactive, digital e-book.

MIS CourseMate also offers many learning features within each chapter, including an Online Case Simulation, a Critical Thinking Challenge, Video Learning Tasks, and a set of Learn It Online activities. To log on to the MIS CourseMate site at **www.cengagebrain.com**, you must create a student account and register this book.

Online Case Simulation: SCR Associates

Session 5: Data and Process Modeling

Overview

The SCR Associates case study is a Web-based simulation that allows you to practice your skills in a real-world environment. The firm offers IT consulting, solutions, and training. SCR plans to open a new high-tech training center, and needs to develop a Training Information Management System (TIMS) to support the operation. You are a newly hired systems analyst reporting to Jesse Baker, systems group manager, and will help her develop the system.

The case study takes you to the SCR Web site, where you receive e-mail and voice mail messages from Jesse, obtain information from SCR's resource libraries, and perform various tasks. Jesse has high standards, but seems very fair. She made it clear that she expects your work to be accurate, thorough, and professional.

Before You Begin

To prepare for this work session, you should review the following topics:

- DFD symbols and diagrams
- DFD context diagrams
- DFD diagram 0
- Decision tables

How Do I Use the Online Case Simulation?

- Read the preview, and review the Chapter 1 background material, if necessary.
- Visit the MIS CourseMate Web site at **www.cengagebrain.com**, locate the **SCR Case Simulation**, and click the **intranet** link. Enter your name and the password **sad10e**.
- When the opening screen displays, select this session. Then check your e-mail and voice mail, and start to work on your task list.

Preview: Session 5

You recently completed requirements modeling tasks for the new Training Information Management System (TIMS). Now you are ready to begin data and process modeling, which will produce a logical model of the new system. You will create DFDs, develop a data dictionary, and use decision tables and decision trees.

Critical Thinking Challenge

In addition to technical ability, IT professionals need critical thinking skills. This feature can help you practice perception, organization, analysis, problem-solving, and decision-making skills that will be valuable in the workplace. You can visit **www.criticalthinking.org** to learn more about critical thinking and why it is so important.

Background

The IT team at Game Technology is moving forward with the new Customer Contact Care information system, or C^3. Your next assignment is to develop a set of data flow diagrams (DFDs). To be sure you can handle the tasks, you decide to review Chapter 5 of your systems analysis textbook. Based on your requirements model, you know that the new C^3 system will have three external entities, with the following data flows:

Entity	Data Flow from the Entity	Data Flow to the Entity
Customer	Contact Information	Sales Specials
Sales Records System	Customer Sales History	None
Marketing Rep	Suggested Contact Plan	Sales Feedback

The C^3 system also will have three processes (**Analyze Customer Data**, **Analyze Sales History**, and **Manage Contact Plan**), a data store (**Customer Profile**), and the following data flows:

Data Flow	From	To
Contact Information	Customer entity	Analyze Customer Data process
Sales Specials	Manage Contact Plan process	Customer entity
Customer Sales History	Sales Records System entity	Analyze Sales History process
Suggested Contact Plan	Marketing Rep entity	Develop Contact Plan process
Sales Feedback	Analyze Sales History process	Marketing Rep entity
Customer Data (two-way)	Analyze Customer Data process	Customer Profile data store
Customer Data (two-way)	Customer Profile data store	Analyze Customer Data process
Profile Data	Customer Profile data store	Manage Contact Plan process
Sales Data	Analyze Sales History process	Customer Profile data store

Practice Tasks

Visit the MIS CourseMate Web site at **www.cengagebrain.com**. Then navigate to the resources for this chapter and locate the **Critical Thinking Challenge** feature. You will complete two Practice Tasks, using what you learned in the chapter. Then you can check your answers to be sure you're ready for the Challenge Tasks.

Challenge Tasks

After you complete the Practice Tasks, you learn about new developments at Game Technology.

Your context diagram and DFD diagram 0 were accurate, but there have been some design changes. Management wants to connect the external Accounts Receivable System directly to the C^3 system as a fourth entity. A two-way data flow called AR Data will connect this entity to the C^3 system. Inside the C^3 system, AR Data will connect to the Analyze Sales History process. Also, another new two-way data flow called Billing Data will connect to the Analyze Sales History process and the Customer Profile data store. To continue, navigate to the **Critical Thinking Challenge** feature for this chapter, select the **Challenge Tasks**, and follow the instructions.

Video Learning Sessions


© craftvision/Stockphoto

The Video Learning Sessions in this chapter explain data and process modeling and decision tables. You'll learn about DFD symbols and diagrams, DFD context diagrams, and DFD diagram 0. You'll also learn how decision tables can provide a graphical model of business process logic.

Before You Begin

To prepare for the tasks, you should review the following topics:

- DFD symbols and diagrams
- DFD context diagrams

How to Watch a Session

Go to the Management Information Systems CourseMate Web site at **www.cengagebrain.com**. Then navigate to the resources for this book, locate the **Video Learning Sessions**, and select a session. You can watch the session on your computer or mobile device, and pause, rewind, or replay a video at any time.

Practice Tasks

After each session, visit the **Your Turn** feature to perform the tasks and check your answers.

Training Tasks

Suppose the IT training manager wants to encourage team members to watch the Video Learning Sessions. To spark interest, she plans to post a set of VLS previews, which you will prepare. Each preview must include an overview of the session, a description of the main topics, and one or more screen examples from the video. To get you started, she provided a task list:

1. Select a Video Learning Session from this chapter. Launch the video and note the title, objectives, and total running time. Include this information in your preview.
2. Run though the session and find a segment that does a good job of explaining a specific skill or concept.
3. In the segment you selected, choose one or more specific screens and capture each screen image with the Windows Snipping Tool. To do this in Windows 7 or Windows Vista, click **Start**, and then type **Snipping Tool**. In Windows 8, you can display the **Start** menu, and then type **Snipping Tool**.
4. Paste the screen image into your preview and explain how it would help a viewer understand the topic. Also add your comments about the video, and why it would be important to an IT professional. Your preview should be well-organized and readable.

Learn It Online

In each chapter, you can use this feature to apply your knowledge and practice your skills. The exercises include Chapter Reinforcement Questions, Flash Cards, Practice Tests, and various games, such as *Who Wants To Be a Computer Genius?, Wheel of Terms, and the Crossword Puzzle Challenge.*