Chapter 13: *State-Transition Diagram*

"Every body continues in its state of rest, or of uniform motion in a right line, unless it is compelled to change that state by forces impressed upon it."

— Sir Isaac Newton
Philosophiae Naturalis Principia Mathematica, Laws of Motion, I , 1687

In this chapter, you will learn:

1. The notation for state-transition diagrams;
2. How to draw partitioned state-transition diagrams;
3. How to build a successful state-transition diagram; and
4. The relationship between STDs and other models.

In the previous chapters, we have seen modeling tools that highlight the *functions* that a system performs, as well as the *stored data* that a system must remember. Now we look at a third kind of modeling tool, the *state-transition diagram* (also known as STD), which highlights the time-dependent behavior of a system.

Until recently, models of a system's time-dependent behavior were important only for a special category of systems known as real-time systems. Examples of these systems (which we discussed very briefly in Chapter 2) are process control, telephone switching systems, high-speed data acquisition systems, and military command and control systems. Some of these systems are passive, in the sense that they do not seek to control the surrounding environment, but rather to react to it or capture data about it. Many high-speed data acquisition systems fall into this category (e.g., a system capturing high-speed scientific data from a satellite). Other real-time systems are more active, in the sense that they seek to maintain control over some aspect of the surrounding environment. Process control systems and a variety of embedded systems fall into this category.

As you might imagine, systems of this kind deal with high-speed external sources of data, and they must provide responses and output data quickly enough to deal with the external environment. An important part of specifying such systems is the description of *what* happens *when*.

For business-oriented systems, this issue has generally not been so important. Inputs may arrive in the system from many different sources and at relatively high speeds, but the inputs can usually be delayed if the system is busy doing something else. A payroll system, for example, does not have to worry about interrupts and signals from external radar units. Typically, the only timing issues that we see in such systems are specifications of response time, which is included in the user implementation model, which we discuss in Chapter 21.

However, we are beginning to see some large, complex business-oriented systems that do have aspects of real-time behavior. If the system is dealing with inputs from thousands of terminals, as well as high-speed inputs from other computer systems or satellite communication facilities, then it may have the same kind of time-dependent issues that a classical real-time system has. Hence, though you may not have to deal with such problems in every system you build, you should be familiar with the modeling tools for time-dependent behavior

13.1 State-transition diagram notation

A typical state-transition diagram is shown in Figure 13.1(a) (though it is somewhat simpler than the diagrams we will see later in this chapter). This diagram shows the behavior of a typical telephone answering machine.

The major components of the diagram are states and arrows representing state changes. There are a variety of alternative notations for state-transition diagrams; one common one is shown in Figure 13.1(b). While it is equivalent in content to Figure 13.1(a) it has the disadvantage of looking too much like a dataflow diagram. To avoid confusion, we will use the notation of Figure 13.1(a) throughout this book.

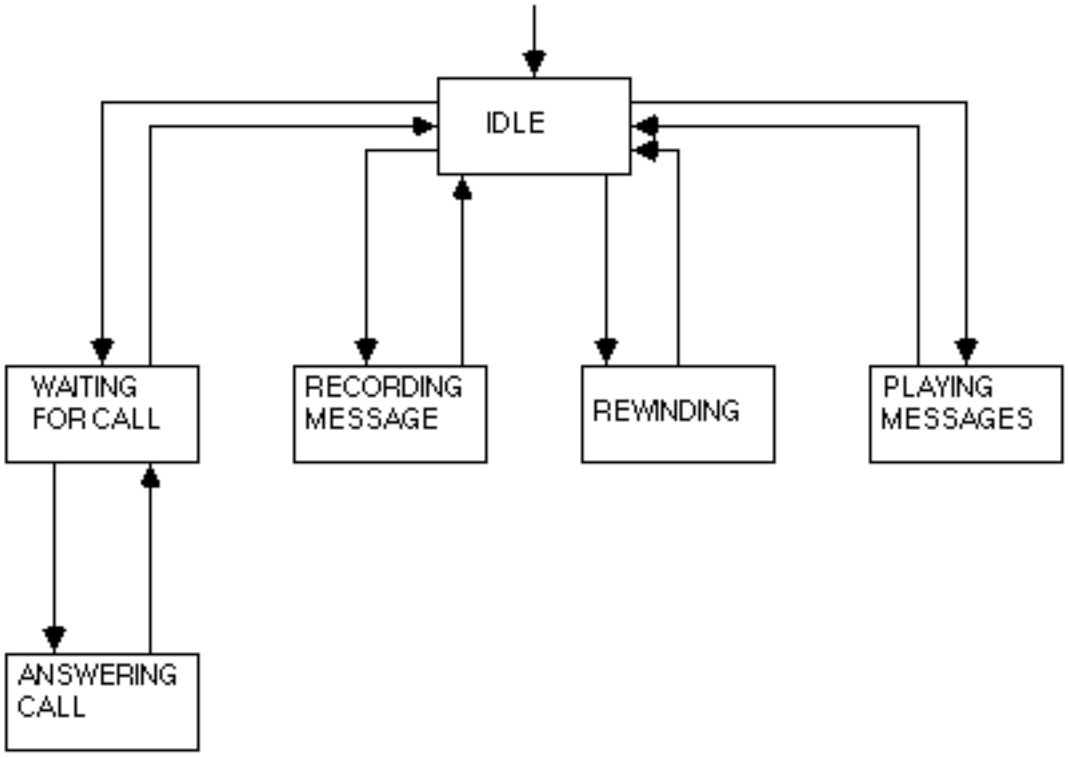


Figure 13.1(a): A typical state-transition diagram

13.1.1 System states

Each rectangular box represents a state that the system can be in. Webster's *New World Dictionary* defines a "state" in the following way:

> A set of circumstances or attributes characterizing a person or thing at a given time; way or form of being; condition.

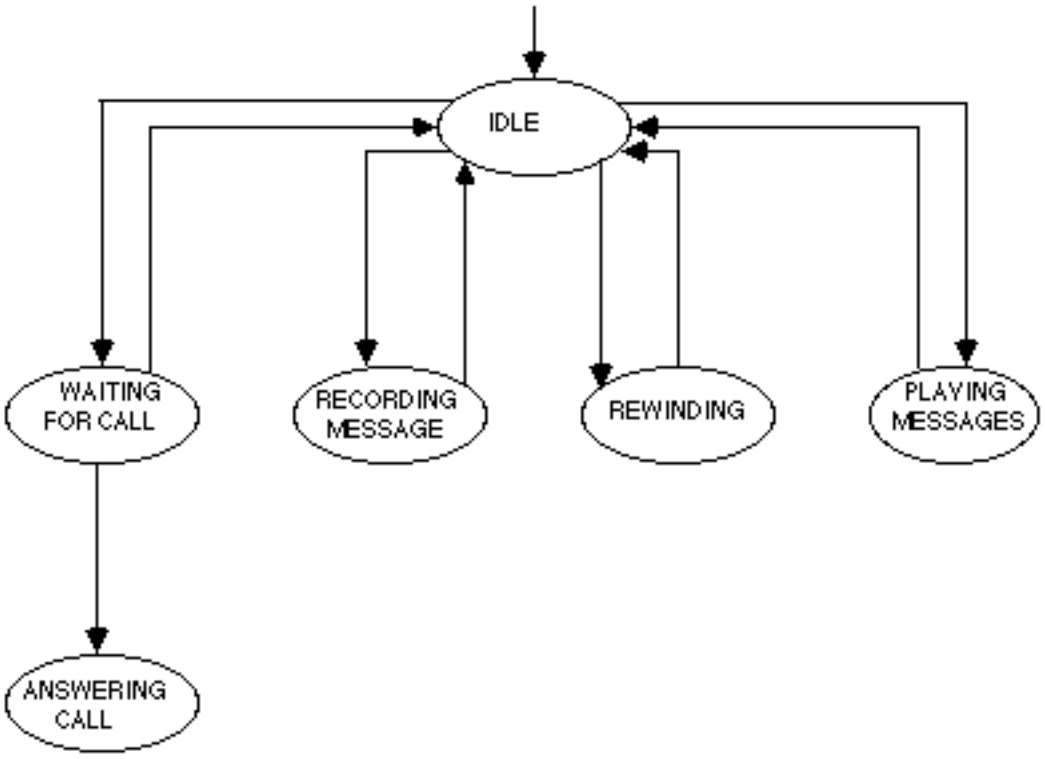


Figure 13.1(b): An alternative state-transition diagram notation

Thus, typical system states might be any of the following:

* Waiting for user to enter password
* Heating chemical mixture
* Waiting for next command
* Accelerating engine
* Mixing ingredients
* Waiting for instrument data
* Filling tank
* Idle

Note that many of these examples involve the system *waiting* for something to occur and are not expressed in terms of the computer *doing* something. This is because our state-transition diagram is being used to develop an essential model of the system **[1]**, a model of how the system would behave if we had perfect technology. One aspect of perfect technology is that our computer operates infinitely quickly, so any processing or computation that the system has to do, or any action that it has to take, will be done in zero time. Thus, any observable state that the system is in can only correspond to periods of time when (1) it is waiting for something in the external environment to occur, or (2) it is waiting for a current activity in the environment (mixing, washing, filling, accelerating, etc.) to change to some other activity.

This does not mean that our systems are incapable of taking action or that we do not intend to show those actions. It's just that actions, which happen instantaneously in our perfect technology model, are not the same as states, which represent observable conditions that the system can be in. Thus, a state represents some behavior of the system that is observable and that lasts for some finite period of time.

13.1.2 Changes of state

A system that existed in only one state would not be very interesting to study: it would be static. Indeed, the information systems that we typically model may have dozens of different states. But how does a system change from one state to another? If the system has orderly rules governing its behavior, then typically only certain kinds of state changes will be meaningful and valid.

We show the valid state changes on our STD by connecting the relevant pairs of states with an arrow. Thus, Figure 13.2 shows that the system can change from state 1 to state 2; it also shows that when the system is in state 2, it can change to either state 3 or back to state 1. However, according to this STD, the system cannot change from state 1directly to state 3. On the other hand, the diagram tells us that the system can change directly from state 3 back to state 1. Note that state 2 has two successor states. This is quite common in STDs; indeed, any one state might lead to an arbitrary number of successor states.

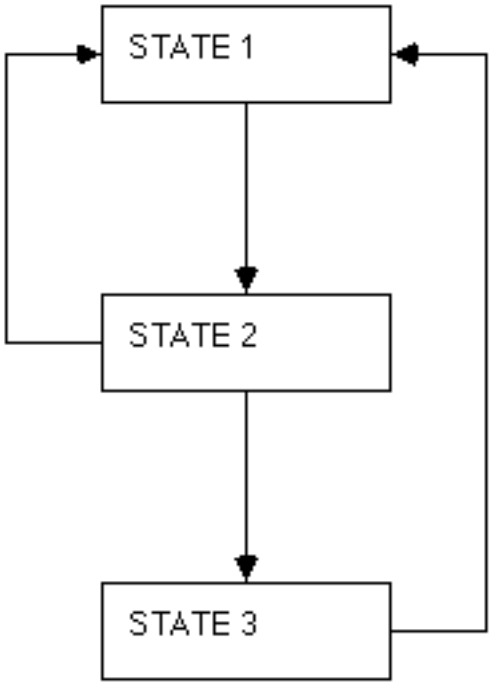


Figure 13.2: Changes of state

While Figure 13.2 gives us some interesting information about the time-dependent behavior of a system, it does not tell us something that may turn out to be very important: what the system's *initial* and *final* states are. Indeed, Figure 13.2 is a steady-state model of a system that has been active forever and will continue to be active forever. Most systems do have a recognizable initial state and a recognizable final state; this is shown in Figure 13.3.

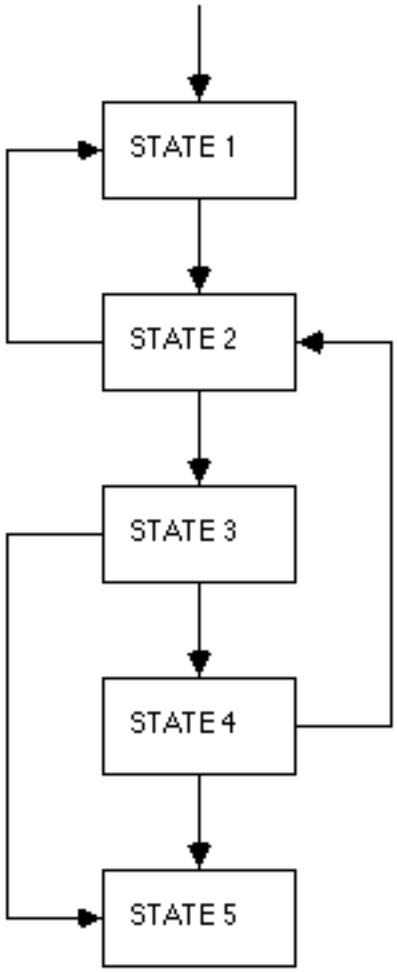


Figure 13.3: Initial and final states

The initial state is typically the one drawn at the top of the diagram, though this is not mandatory; what really identifies state 1 in Figure 13.3 as the initial state is the "naked" arrow that is not connected to any other state. Similarly, the final state is often the one drawn at the bottom of the diagram, but this is not mandatory. What really identifies state 5 as the final state is the absence of an arrow leading out of state 5. In other words, once you get to state 5, you aren't going anywhere!

Common sense tells us that a system can have only one initial state; however, it can have multiple final states; the various final states are mutually exclusive, meaning that only one of them can occur during any one execution of the system. Figure 13.4 shows an example in which the possible final states are states 4 and 6.

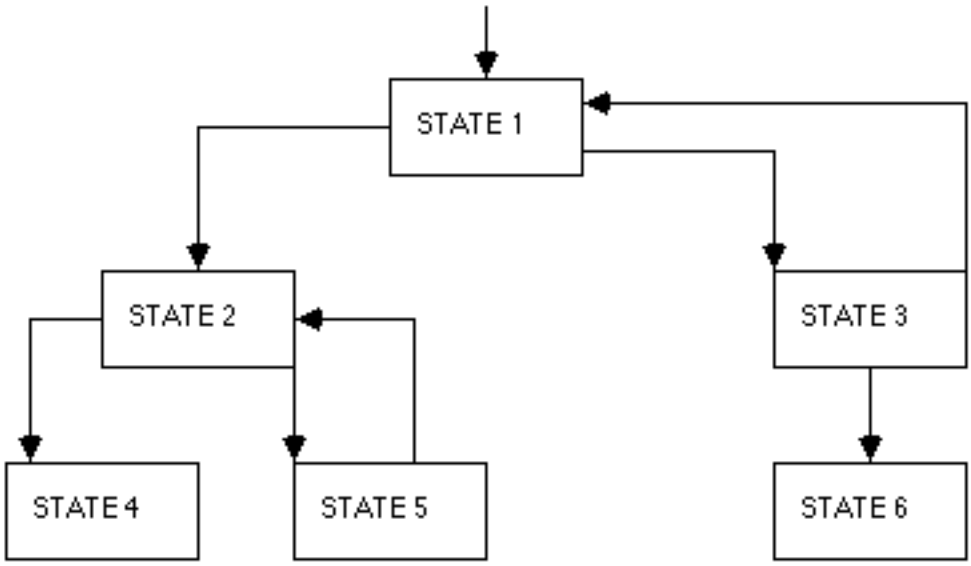


Figure 13.4: Multiple final states of a system

Since we are using STDs to build an essential model, we also assume that state changes occur instantaneously; that is, it requires no observable time for the system to change from one state into another state. When the designers and programmers begin to build an *implementation model*, this will be a real issue: it typically *does* take a few microseconds for a computer to switch from one processing activity to another, and they must ensure that it happens quickly enough that the environment does not get out of control.

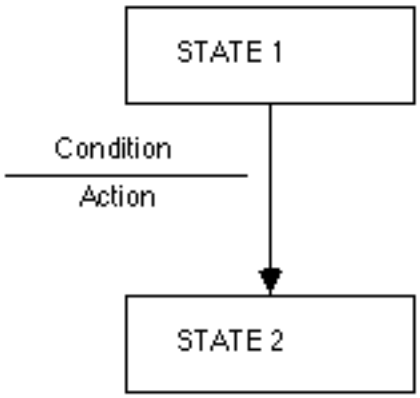


Figure 13.5: Showing conditions and actions

13.1.3 Conditions and actions

To make our state-transition diagram complete, we need to add two more things: the *conditions* that cause a change of state, and the *actions* that the system takes when it changes state. As Figure 13.5 illustrates, the conditions and actions are shown next to the arrow connecting two related states.

A condition is some event in the external environment that the system is capable of detecting; it will typically be a signal, an interrupt, or the arrival of a packet of data. This will typically cause the system to change from a state of waiting for X to a new state of waiting for Y, or carrying out activity X to carrying out activity Y.

But as part of the change of state, the system will typically take one or more actions: it will produce an output, display a message on the user's terminal, carry out a calculation, and so on. Thus, actions shown on the STD are either responses sent back to the external environment, or they are calculations whose results are remembered by the system (typically in a data store shown on the dataflow diagram) in order to respond to some future event **[2]**.

13.2 Partitioned diagrams

In a complex system, there may be dozens of distinct system states; trying to show them all on a single diagram would be difficult, if not impossible. Thus, just as we used leveling or partitioning with dataflow diagrams, we can use partitioning with STDs. Figure 13.6(a) shows an example of two levels of state-transition diagrams for a complex system.

Note that in this case, any individual state of a higher-level diagram can become the *initial* state for a lower-level diagram that further describes that higher-level state; and the final state(s) in a lower-level diagram correspond to the exit conditions in the associated higher-level state. In other cases, the systems analyst may need to show, explicitly, how a low-level STD diagram exits to an appropriate place in the higher-level diagram.

An example of the need for a partitioned state-transition diagram might be the automated teller machine now found in most banks; an STD for this application is shown in Figure 13.6(b).

13.3 Building the state-transition diagram

Now that we have seen the notation for state-transition diagrams, we briefly discuss the steps in building one. You can follow either of two approaches:

1. You can begin by identifying all possible system states, representing each one as a separate box on a sheet of paper. Then you can explore all meaningful connections, (i.e., state changes) between the boxes.
2. Alternatively, you can begin with the initial state, and then methodically trace your way to the next state(s); and then from the secondary state(s) to the tertiary state(s); and so on.

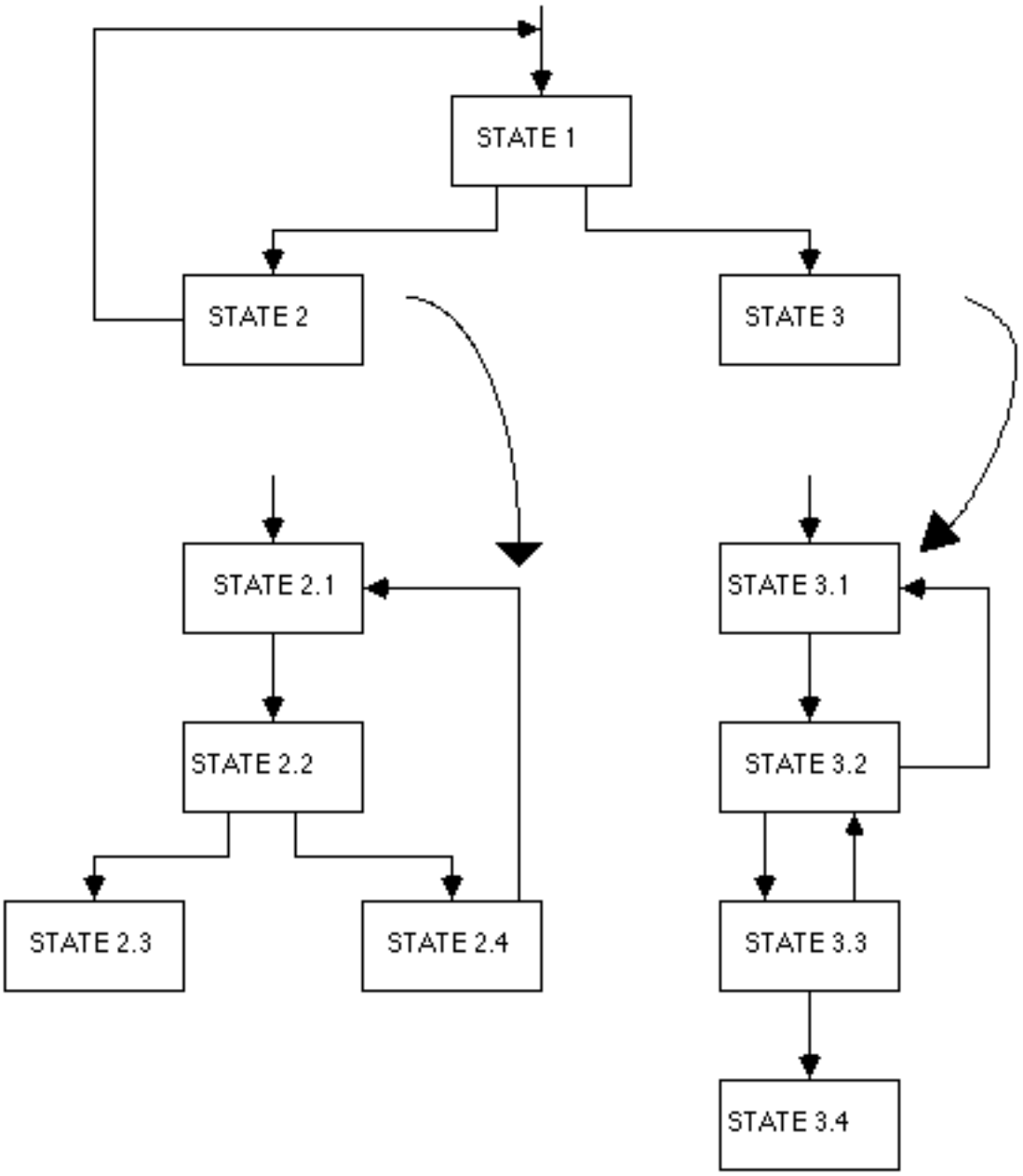


Figure 13.6(a): Two levels of STD

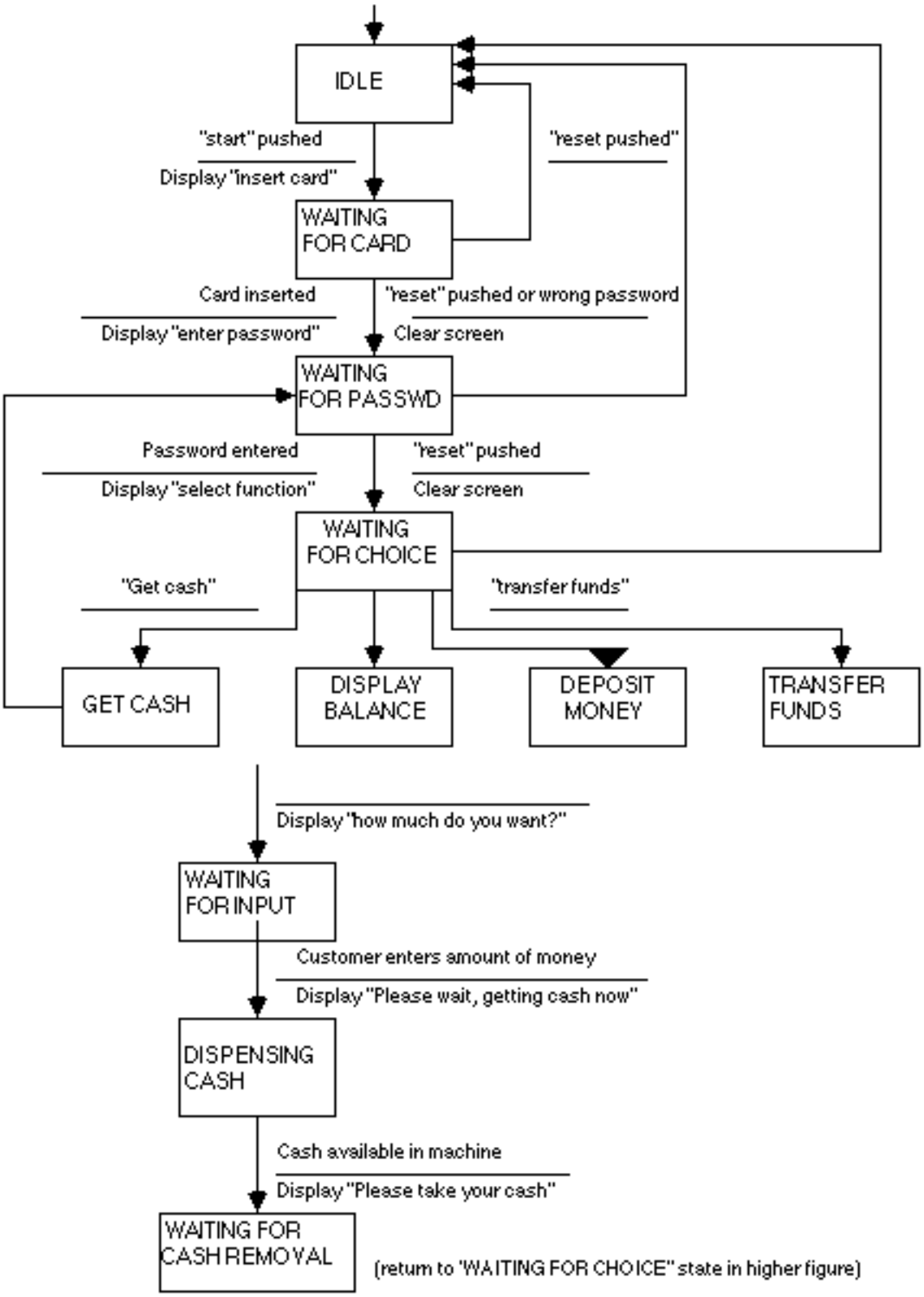


Figure 13.6(b): A partitioned STD for an ATM machine

Your approach will be dictated, in many cases, by the user with whom you are working, particularly if the user is the only one familiar with the time-dependent behavior of the system.

When you have finished building the preliminary STD, you should carry out the following consistency checking guidelines:

* *Have all states been defined?* Look at the system closely to see if there is any other observable behavior or any other condition that the system could be in besides the ones you have identified.
* *Can you reach all the states?* Have you defined any states that do not have paths leading into them?
* *Can you exit from all the states?* As mentioned above, the system may have one or more final states with multiple entrances into them; but all other states must have a successor state.
* *In each state, does the system respond properly to all possible conditions?* This is the most common error when building a state-transition diagram: the systems analyst identifies the state changes when normal conditions occur, but fails to specify the behavior of the system for unexpected conditions. Suppose the analyst has modeled the behavior of a system as shown in Figure 13.7; she expects that the user will press a function key on his terminal to cause a change from state 1 to state 2 and a *different* function key to change from state 2 to state 3. But what if the user presses the same function key twice in a row? Or some other key? If the system behavior is not specified, there is a good chance that the designers and programmers will not program for it either, and the system will exhibit unpredictable behavior under a variety of circumstances.

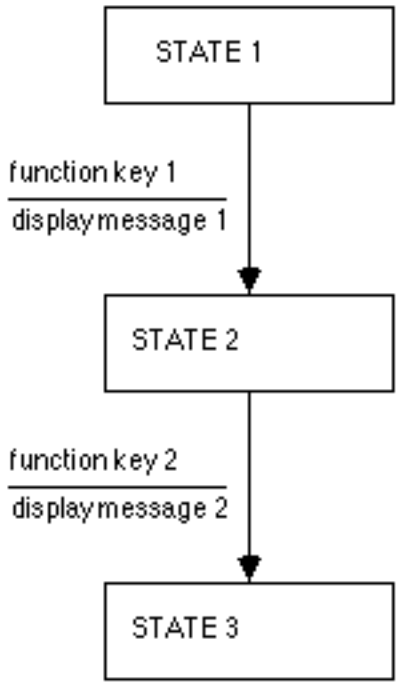


Figure 13.7: An incomplete STD

13.4 The relationship to other model components

The state-transition diagram can be used as a modeling tool by itself. However, it can be, and usually should be, used in conjunction with those other tools.

In most cases, the state-transition diagram represents a process specification for a control bubble in a dataflow diagram. This is illustrated in Figure 13.8; note that the conditions in the STD correspond to *incoming* control flows on the DFD, and the actions on the STD correspond to *outgoing* control flows on the DFD. As a high-level modeling tool, the state-transition diagram can even serve as a process specification for the entire system. Thus, if we represent the *entire system* with a one-bubble dataflow diagram **[3]**, we can use a state-transition diagram to show the sequence of activities within the system.

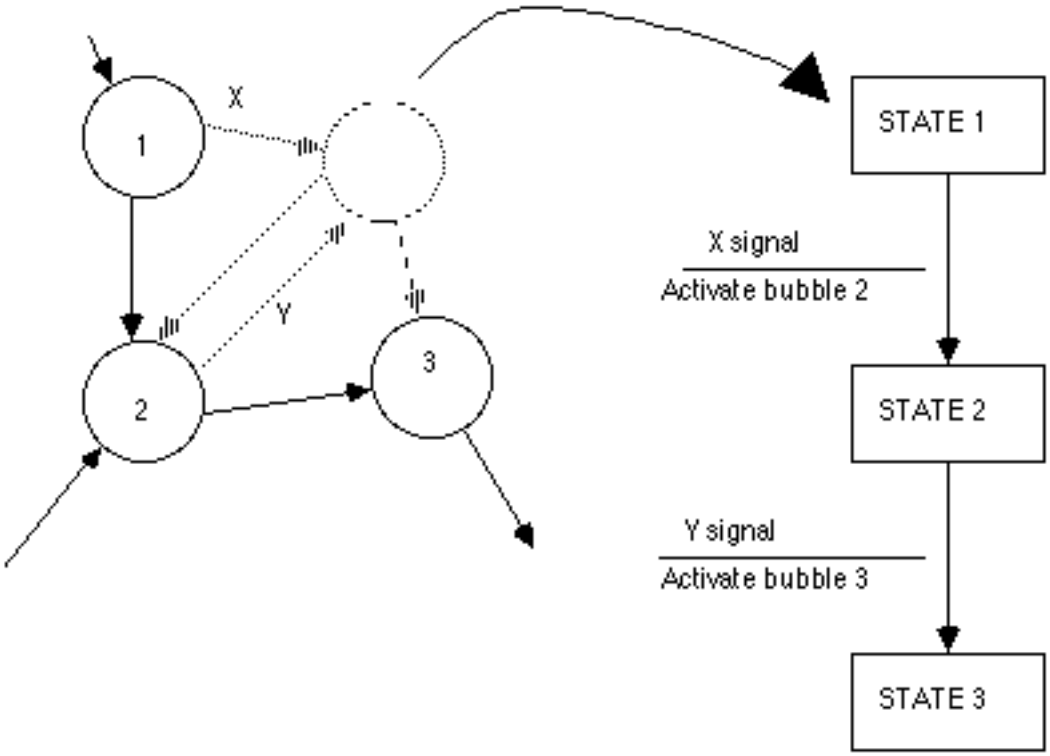


Figure 13.8: Relationship between DFD and STD

13.5 Summary

The state-transition diagram is a powerful modeling tool for describing the required behavior of real-time systems, as well as the human interface portion of many on-line systems. Though it is not as widely known or used in the development of business-oriented information systems, it is a tool that you should become familiar with, because, in the future, we can expect that more and more systems, whether business-oriented or scientific/engineering in nature, will take on real-time overtones.

References

1. *Webster's New World Dictionary*, Second College Edition. New York: Simon & Schuster, 1980.

Endnotes

1. We will discuss the concept of an essential model in more detail in Chapter 17.

2. Note that to carry out an action the system may require additional inputs from the external environment. Thus, we can say that each condition corresponds to an external event to which the system must respond, and that those external events will usually be recognized by the system when some incoming dataflow arrives. However, it is not necessarily true that every incoming dataflow to the system is an event corresponding to a condition on the STD.

3. Such a diagram is known as a context diagram. We will discuss the context diagram in more detail in Chapter 18.

Questions and Exercises

1. What is a state-transition diagram? What is its purpose?

2. What kind of system is most likely to use an STD as a modeling tool?

3. Are STDs important tools for describing the requirements of a typical business-oriented information system? Why or why not?

4. Are STDs important tools for describing the design/implementation of a typical business-oriented information system? Why or why not? If so, what kind of business systems?

5. What are the two major components of an STD?

6. Show an alternative notation for an STD, that is, one different than the standard diagrams shown in this chapter and throughout the book.

7. What is the definition of a state?

8. Give three examples of a state.

9. What is a change of state? How is it shown on an STD?

10. What is a successor state?

11. What is the initial state of a system? How many initial states may there be in a system?

12. What is the final state of a system? How many final states may there be in a system?

13. What are conditions in an STD? How are they shown?

14. What are actions in an STD? How are they shown?

15. How many conditions can there be in a state-transition diagram?

16. How many actions can be associated with each conditions in a state-transition diagram?

17. Which of the following sound like reasonable states? For those that do not sound like reasonable states, indicate why:

(a) Compute sales tax
(b) Monitoring reagent mixture
(c) Customer-billing-address
(d) Product-file
(e) Elevator rising
(f) Reagent temperature out of range
(g) Update invoice total
(h) Stop elevator
(i) Interrupt key pressed
(j) Processing data

18. What is a partitioned state-transition diagram?

19. What is the relationship between initial states and final states in a partitioned STD?

20. How many levels can there be in a partitioned STD?

21. What are the two common approaches for building an STD?

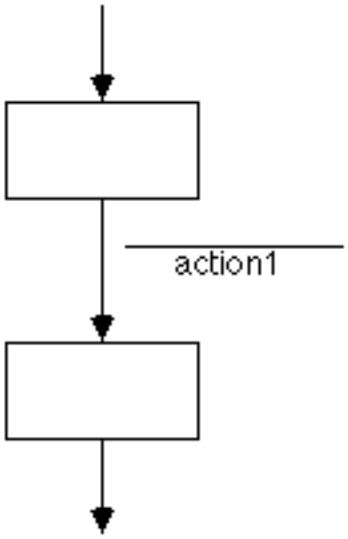
22. What are the four guidelines for determining whether an STD is consistent?

23. What is the relationship between an STD and a DFD?

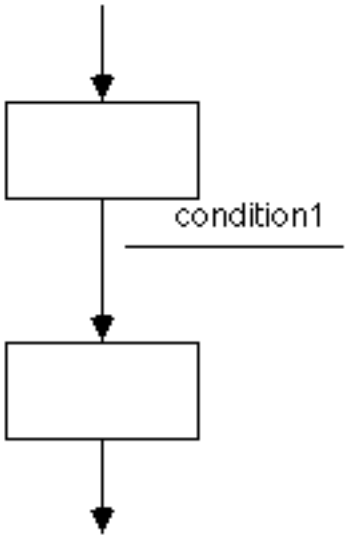
24. What is wrong with the following STD?

25. What is wrong with the following STD?

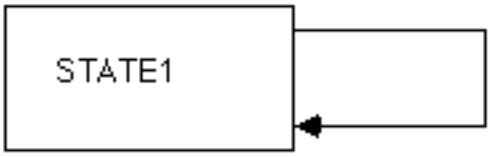
26. What is wrong with the following STD?



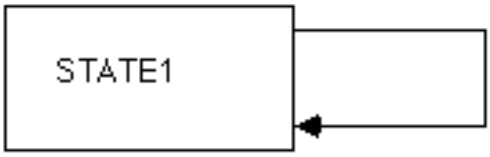
27. What is wrong with the following STD?



28. What is wrong with the following STD?

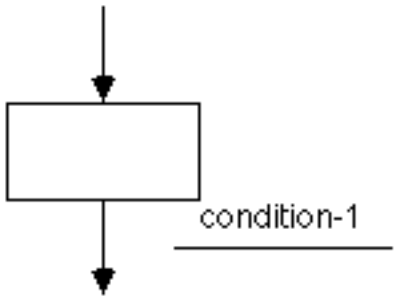


29. What is wrong with the following STD?

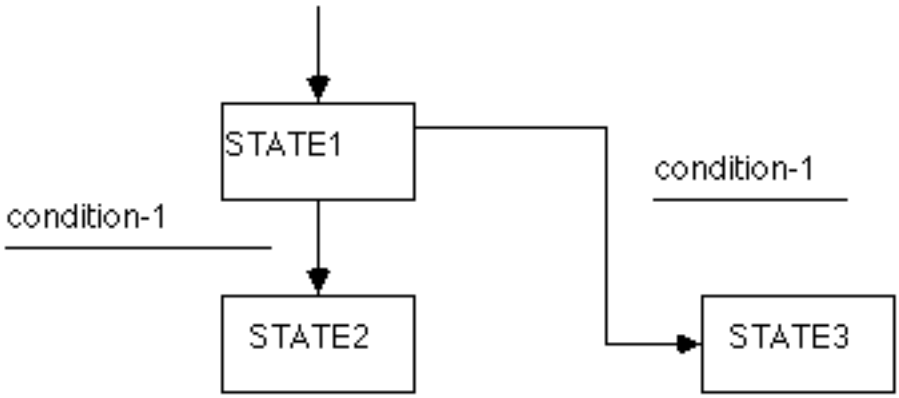


Questions and Exercises (cont.)

30. What is wrong with the following STD? If you do not think anything is wrong with it, describe what might be happening in the system being modeled by this STD.



31. What is wrong with the following STD?



32. In a complex system, should the systems analyst begin by drawing a set of DFDs for the system, or begin with ERDs, or begin with STDs?

33. Where should the details of STD conditions and actions be described in a system model?

34. Draw a state-transition diagram for a simple tape recorder or cassette tape player.

35. Draw a state-transition diagram for your bank's automated teller machine.

36. Draw a state-transition diagram for a digital wristwatch (most digital watches these days have a "normal" mode, as well as an alarm clock and a chronograph).

37. Draw a state-transition diagram for a microwave oven.

38. Draw a state-transition diagram for the human interface menu for Microsoft Excel.