CHAPTER

Business Process Models

5

Business process modeling is the third of the four business modeling disciplines we describe in this book. A business process model describes tasks and the ordering of these tasks: what work is performed and when it is performed. A business process model also captures who performs the tasks. This chapter explains business process models.

Some businesses run smoothly. Each task is a part of an elegant dance, with employees doing just what they need to do and working together to make a simple and beautiful whole. Other businesses run rough. They get their work done, but every task is a heroic struggle. Chaos reigns. Business processes are the difference between the smooth businesses and the rough ones. The smooth businesses execute good business processes; the rough businesses execute poor ones.

A *business process* is a collection of step-by-step tasks that a business uses when it performs its work. For example, all restaurants pursue the same goals of serving food for hungry customers, but they differ in the details of their business processes. They greet customers differently, they take reservations differently, and they prepare dinners differently.

Now that Cora Group has been successfully integrated into Mykonos, you are charged with growing the revenues of the Cora Group restaurants. You learn that Portia faces a business process problem. Portia's customers are experiencing long delays before they are seated. You are concerned that the delays will result in a poor customer experience and will lead to reduced revenue. You would like to investigate what is happening in this restaurant: Why are customers waiting? Is there an issue with the way Portia takes reservations and assigns tables? Can the problem be solved with a new reservation system? With pagers? With more servers? With more efficient seating arrangements?

You want to understand the business process of Portia customer dining, from the beginning to the end, from the time a dining party arrives until they leave the restaurant. You want to know how customers are greeted, how reservations are handled, how customers are seated, how they are served, and how tables are

freed and cleaned so that new customers can be served. You also want to understand who is interacting with the customers along all these steps and how their jobs are performed.

To achieve this understanding, you create a *business process model*. A business process model is a model of a business process—a model of what work is being done and who does it.

WHY MODEL BUSINESS PROCESSES?

Why do we care about modeling business processes? As you will recall from Chapter 1, business models in general are used for eight purposes: communication, training and learning, persuasion and selling, analysis, managing compliance, as requirements for developing software, executing directly as software, and knowledge management and reuse.

All eight purposes apply to business process modeling. Some businesses build business process models as part of their transformation initiatives to capture the way they perform their work today and the way they will perform work in the future. These models are used to *communicate* to the employees what will change and how the change will affect their day-to-day work lives. Sometimes models are used to *train* new employees so that they understand all the tasks they are expected to perform and the order in which they should perform them.

Process models are often *analyzed*. One business process is compared with others to see which process is best. Analysis helps us understand the cost involved with each process, how many people are needed, and where delays occur. Such analysis can also be used to *persuade.* If we think that outsourcing a business function is cheaper than keeping the function in-house, we can show process models with the function in-house and with the function outsourced and demonstrate the difference in cost. Sometimes process models are used to persuade clients or prospective clients—for example, to persuade a client that we understand his business and his challenges.

Process models are useful in *managing compliance* with a new regulation. By modifying an existing process (or by implementing a new process) we ensure that we are complying with a regulation. We can investigate the way we are doing work today and compare it to the work that needs to be accomplished to achieve compliance.

Business process models can provide us with information useful in capturing *software requirements*. By capturing the way users perform the work, we can understand their needs. We can investigate each activity in a process and determine whether the activity is supported by a software application today and whether it should be supported by an application in the future. We can trace the software requirements of the future applications back to the activities they support. (Chapter 12 explores the relationship between business process activities and software requirements.)

Business processes can be *executed as software*. A business process executed as software becomes workflow. A user is presented with user interfaces that walk

her through the steps she must perform. To execute a business process as workflow, a specialized tool must be used to convert from the business process to code that can be executed in a business process engine. The tool then ensures that the modeled workflow is realized and followed. Execution of business processes as workflow is explained in Chapter 12.

When an organization practices *knowledge management*, it applies knowledge gleaned in one part of the organization to another part of the organization. Often this knowledge includes how to perform a business process. Business process models capture how work is performed and who performs it. They also show how a person interacts with others—both others within their organization and others external to it.

ACTIVITIES

People in organizations perform work. For instance, help desk employees handle incoming customer calls. An accountant updates a company's balance sheet. A distributor ships an ordered product. And in the Mykonos restaurants, a restaurant host takes reservations and seats parties of diners. Business process modeling is about modeling work that is being performed—modeling what people do, how they do it, and the activities they perform along the way.

What is an activity? An *activity* is a discrete chunk of work, something with a beginning and an end, that is performed one or more times. For example, **Serve Appetizers** is an activity. At every Mykonos restaurant, the activity **Serve Appetizers** is performed many times every evening.

Typically an activity is one step of a larger business process. For example, **Serve Appetizers** is part of the larger business process **Serve Meal**. There are other activities within **Serve Meal**, including **Serve Entrees** and **Serve Desserts**.

Every activity performs work. For example, when a call is made to a help desk, a help desk support person starts by opening a trouble ticket and by asking the caller for his personal information—his name and phone number. **Open Ticket** is one activity in this process, and **Get Caller Info** is another. The incoming call itself is not an activity, since no work is performed. Instead the call is a trigger for the first help desk activity, **Open Ticket**. The ticket itself is also not an activity; it is a record that is created by **Open Ticket**.

Figure 5.1 shows the activity **Welcome Diner**, performed by the restaurant host. Figure 5.1 is atypical; activities rarely appear by themselves. Instead,

FIGURE 5.1 An activity

FIGURE 5.2 Three related activities

several activities typically appear together as part of a larger process, as in Figure 5.2. Once the host welcomes the diners, she asks for a name and whether they have a reservation. The host then checks the information against the reservation records to see whether the diners indeed have a reservation.

SEQUENCE FLOWS

A sequence flow is a connection between two activities, showing that one activity is performed before the other. A sequence flow is shown as a solid line with an arrow, from the activity performed first to the activity performed next. In Figure 5.2 there is a sequence flow between the activity **Welcome Diner** and the activity **Get Diner Information**, and another sequence flow between **Get Diner Information** and **Check Reservations**. First the activity **Welcome Diner** is performed. When this activity finishes, the next activity, **Get Diner Information,** begins. After the **Get Diner Information** finishes, the activity **Check Reservations** begins. When this last activity finishes, the process is complete.

This process is used to greet many parties that are arriving at different times. The host might be checking on the reservations of one party before welcoming another party. However, for a particular party, the sequence of activities that are performed is clear and definitive. First the party is welcomed, then the party information is collected, and then reservations are checked.

As you will recall from Chapter 4, a business organization model can include an interaction between two organizations, showing that they work together. An interaction is depicted much like a sequence flow. Both are arrowed lines from one model element to another. But a sequence flow has a very different meaning than an interaction. An interaction is an association between two organizations (or two roles, or an organization and a role). A sequence flow is an association between two activities. An interaction means that the two organizations work together. A sequence flow means only that one activity occurs after the other. An interaction is labeled by the deliverable that one organization delivers to the other. A sequence flow is usually unlabeled and is never labeled with a deliverable.

ACTIVITY ATTRIBUTES

Every activity has *attributes*, which capture details of the work. Activity names are short, typically no longer than four words, and better when they are two or three words. **Check Reservations** is a good name; it is simple and easy to understand. The name of an activity need not convey the details of the way the activity is performed. From "check reservation" we do not know how the reservations are checked, whether there is a reservations system or a big leather-bound book. Rather, the name simply describes the work being performed.

The description of an activity gives more detail about the work, what it means, and how it is performed. For example, a description for **Check Reservations** states:

> *Check the reservation book to see whether the reservation exists. Verify that the party arrived before the reservation time.*

A description typically notes whether a software application is used to perform the activity. If an application is used, the description includes the way the person interacts with the application. For example, if the restaurant has a reservations system instead of a book, the description for **Check Reservations** is instead:

> *Use the reservation system to check whether the reservation exists, searching for the reservation by name or by time. Verify that the party is not late, that they have arrived before the reservation expires.*

Descriptions should have at least a sentence to describe the activity, and two or three sentences are better. Descriptions are an excellent place to capture subject matter expertise.

Activities are temporal. Each activity takes time to complete. Some activities are fast, taking seconds. Other activities are slow, taking months. Often there are delays—delays before the work, delays during the work, and delays because of the work. It is important for subsequent process analysis to capture these times—both the work times and the delay times. If the business process is simulated, the activity times are used by the simulation engine. (Business process simulation is described in Chapter 11.)

Typically an activity is performed by a person, the person who does the work of the activity. This person is called the activity's *resource.* Of course different people perform the same activity at different times. Jessica might greet people today and Austin tomorrow. So the resource of an activity is not a single person but a role. As you recall from Chapter 4, a role is the responsibility a person assumes when he holds a position in an organization. For example, the role **Host** is the resource of the activity **Check Reservations**. When a particular party arrives, their reservations are checked by a single person who plays that role.

People do not work for free. Every resource has a cost, and the cost varies from role to role and from person to person. Details about costs of the resources are useful to understand the end-to-end cost of a process.

Some activities are *manual* work, performed by a resource without any assistance. When a host welcomes an arriving party, she does so without the assistance of any technology. On the other hand, some activities are supported by technology. When a host checks for the party's reservations, she looks up their name in a reservations system. In analyzing a business process model, it is useful to understand what work is performed with the assistance of a system. We can then analyze how new technology could be used and how the activity's resource would interact with that new technology in performing the activity.

Some activities are *solely software*, performed entirely by a software application, with no person involved. For example, Mykonos orders staples for all the restaurants—mineral water, olive oil, and cleaning supplies. These restaurant staples are ordered automatically whenever inventory is low, without anyone involved in the ordering activity. In this solely software activity, the application does not support a (human) resource who is performing the activity. Instead the application is the sole resource performing the activity.

It is occasionally useful to model solely software activities as part of a larger business process, but most activities involve people. Business process modeling is not about modeling software; it is about modeling the work that people do. Solely software activities are uncommon in good business process models.

EVENTS

A business process has a beginning and an end. A process begins with a *start event* and ends with an *end event*. All the activities of the business process—the actual work performed—occur after the start event and before the end event.

When the restaurant host welcomes an arriving party, the welcoming is the first activity. However, to greet the party, something must have happened: the party walked into the restaurant. This arrival is what starts the first activity and is the trigger for everything that happens afterward. That arrival is a start event. A *start event* is something happening that begins a business process. **Diner Arrives** is the start event of the restaurant dining process, as shown in Figure 5.3.

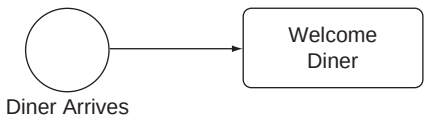

FIGURE 5.3 A start event

FIGURE 5.4 A simple end-to-end process

Note the name of the start event **Diner Arrives** sounds different from the names of the activities, such as **Welcome Diner**. Activity names are typically imperative sentences; they sound like commands. The verb is at the beginning of the name. The name of a start event is typically a declarative sentence, describing something that happens. The verb is at the end.

The *end event* of a process is when the process ends, after the last activity. Sometimes the process has a natural end, and sometimes we end a process because we are not interested in modeling anything beyond. Our modeling scope determines where we end. For example, suppose we care only about how diners are greeted. Then our process is simple, as shown in Figure 5.4. We have a start event to show the diner arriving; we have activities to greet the diner and to check reservations, and we end with **Diner Seated**.

But suppose instead that we intend to capture many more activities and finish only when the party leaves the restaurant. In that situation our restaurant dining process will be much larger than the simple process shown in Figure 5.4.

Most processes have multiple end events. Diners might leave after eating and paying, or they might leave early, disgruntled by long delays in their restaurant experience. They might even leave before they are seated, after waiting too long for a table, or because they are called out to perform emergency surgery. A process can also have multiple start events, showing different ways that work begins.

Some processes have an *intermediate event*, an event that happens after the process starts but before it ends. Many intermediate events model delays. For example, when the first person of a large party arrives at Portia, she waits. Portia's policy is to seat a party only when everyone is present. When the first diner arrives, the host checks the reservations, but she does not seat the diner until the rest of the party arrives. Figure 5.5 shows the process with the intermediate event **Party Arrives** after **Check Reservations**. The diner waits at **Party Arrives**

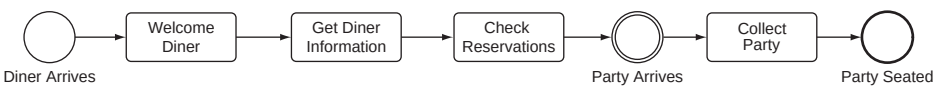

FIGURE 5.5 A process with an intermediate event

FIGURE 5.6 A message start event and a timer start event

until everyone is present. Once the rest of the party arrives, the host collects the party and they are seated.

Like start events, the name of an intermediate event is also a declarative sentence. For example, the intermediate event in Figure 5.5 is **Party Arrives**, much like the name of the start event, **Diner Arrives**.

Events of all three varieties can have descriptions, just as an activity can have a description. For example, the description of **Party Arrives** explains the Portia policy of making the party wait until everyone has arrived. (Alternatively, the policy might be modeled as a business rule, as described in Chapter 6.)

Events also support other attributes. Start events record detail about when work starts. For example, **Diner Arrives** includes attributes modeling how often dining parties arrive, how many on which night of the week, the sizes of the parties, and so on. These attributes are used for process simulation, as described in Chapter 11. Intermediate events have similar attributes about how long work is delayed.

What triggers a start event? Some start events are triggered by the arrival of a message from elsewhere. For example, a customer order at a catalog retailer begins when an order arrives from a customer. This start event begins with the receipt of a message. A start event with a *message trigger* is depicted graphically as a little envelope within the start event, as shown on the left of Figure 5.6.

Some start events are trigger by a temporal cycle—something that happens every night or every month. For example, the complete cleaning of Portia starts every night at midnight. Temporally started events are said to have a *timer trigger*. A start event with a timer trigger is depicted with a little clock, as shown on the right of Figure 5.6.

An intermediate event can also have a timer trigger if it happens at a particular time, or a message trigger if it sends or receives a message. There are other useful event triggers, including process cancellation, process termination, error condition, and aborted transaction. These event triggers are described later in this chapter.

LANES

A business process model graphically shows who performs which activities. Each role that performs activities in a business process has a *lane*—a horizontal stripe like a lane in a swimming pool. Figure 5.7 shows a process for collecting a

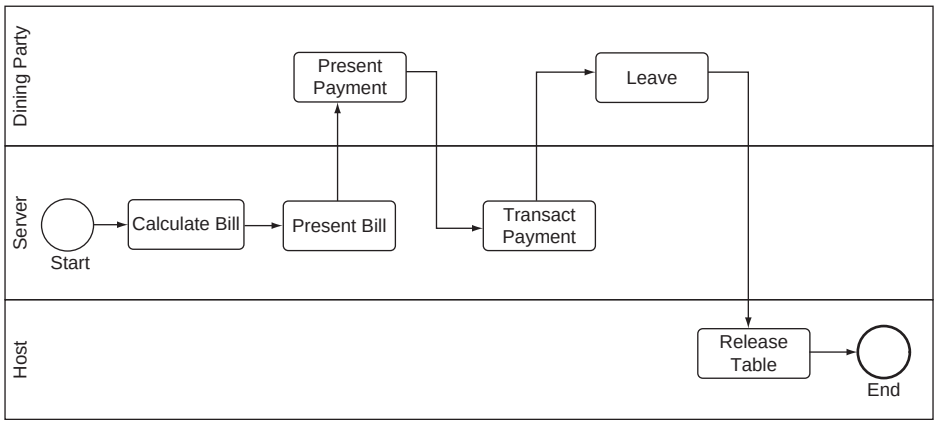

FIGURE 5.7 Lanes in a business process

payment from a departing dining party. Figure 5.7 has three lanes: one for the dining party, one for the server, and one for the host. The dining party performs the activities in the **Dining Party** lane: **Present Payment** and **Leave**. Similarly, the server performs the activities in the **Server** lane, and the host performs the activity in the **Host** lane. Each lane is named by the role or organization who performs the work. This role (or organization) is called the *participant* of the lane.

Some activities have an outgoing sequence flow to another activity in the same lane, when the same person performs the next activity. For example, the activity **Calculate Bill** is followed by **Present Bill**, both performed by the server. Some activities have an outgoing sequence flow to an activity in another lane, when the next step is performed by a different role. For example, after the diner leaves in **Leave**, the host releases the table in **Release Table**.

In Figure 5.7, the participant of the top lane is **Dining Party**, the customer of this process. Figure 5.7 is typical. When a customer is shown, the customer is usually the top lane. The other participants who serve the customer are situated in lanes below. Putting the customer on top is a common convention that makes business process models easier to read.

Not every process model has a customer on top. Sometimes a model is less visually complex if the customer lane is in the middle, neither on top nor on the bottom. And sometimes a process is completely concerned with internal matters and has no participant who can be called a customer. But these exceptions to the customer-on-top convention are not so common. In most models, the customer is on top.

When a process model has more than two lanes, the modeler must decide which lane is placed where. Should the **Server** lane be above the **Dining Party** or below? A good rule of thumb is to place lanes to minimize sequence flow.

When a sequence flow is from an activity in one lane to an activity in another lane, it is better if the two lanes are adjacent so that the sequence flow is short. If the two lanes are distant from each other—e.g., separated by three other lanes—the sequence flow is long. It is harder to read the resulting diagram.

GATEWAYS

Our processes so far follow a single path, one activity at a time, from the start event to the end event. For example, in Figure 5.4 the host finds the reservation, a table is available, and the diner is seated. In reality work is always more complex: conditions arise that cause the sequence flow to diverge, either to one sequence flow or to an alternative. The diners either have a reservation or they do not. Different activities occur depending on whether they have a reservation. Similarly, the diners order appetizers or they do not.

We use a *gateway* to model sequence flow alternatives. A gateway is depicted as a diamond shape. Multiple sequence flows exit a gateway. The actual sequence flow taken in a particular situation depends on the condition modeled by the gateway.

The business process fragment in Figure 5.8 shows what happens after the host checks reservations. There are two alternative outcomes of the reservations check: either the party has reservations or they do not. These two outcomes are shown as two outgoing sequence flow of the gateway **Reservation?** If they have a reservation, the host assigns a table in **Assign Table**. If the party has no reservation, the customers are turned away in **Turn Customers Away**.

Gateways are named. The name of a gateway is a question, with the alternative answers to the question as labels on the outgoing sequence flows. In Figure 5.8,

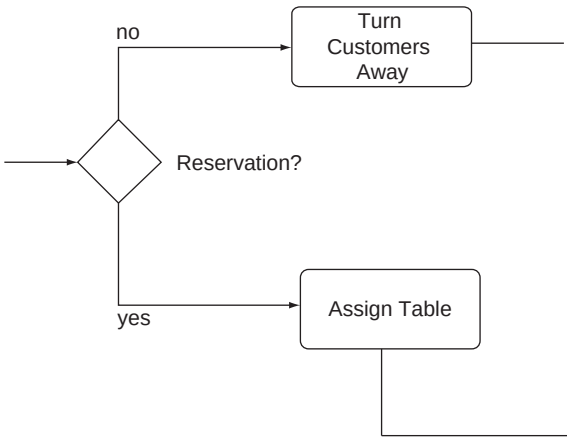

FIGURE 5.8 A gateway

the gateway is **Reservation?** and the two outgoing sequence flows are labeled **no** and **yes**.

Of course, Portia turns customers away only as a last resort. Instead the host checks whether a table is available. If a table is available, the party is seated anyway, as though they had a reservation. The process fragment Figure 5.9 shows this more accurate process, with a second gateway **Seat Now?** showing the result of the table availability.

Figure 5.10 shows more of the same process. If the party has no reservations and no table is available, the host looks for a way to arrange tables. If no such rearrangement exists, the party will wait for a table, perhaps waiting 20 minutes or an hour. Ideally the party simply waits until a table is available, but in practice many parties are impatient and ask periodically about the status of their table. So the process in Figure 5.10 shows the host checking for availability after a wait, only to sometimes ask the party to wait some more. The activities **Check Availability, Check for Rearrangement**, and **Wait for Table** are part of a *sequence flow loop*, a cycle of activities connected by sequence flow.

Some gateways model decisions, where someone decides which sequence flow to take. **Seat Now?** in Figure 5.10 is such a decision. The host decides whether to seat the party now or whether to check for a rearrangement of tables. The decision is not difficult or time-consuming; one can hardly imagine the host laboring over this simple choice on a busy Saturday evening. But it is a decision nonetheless.

As we describe in Chapter 6, gateways that model decisions can be guided by one or more business rules. The decision **Seat Now?** is guided by a business rule that describes what to do when a table is available. Other business rules provide further guidance in other situations—for example, when the party includes regulars, friends of the owner, or celebrities.

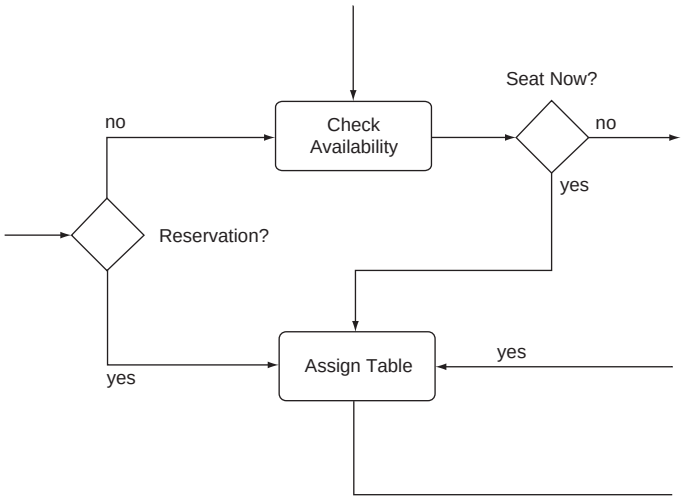

FIGURE 5.9 Two gateways

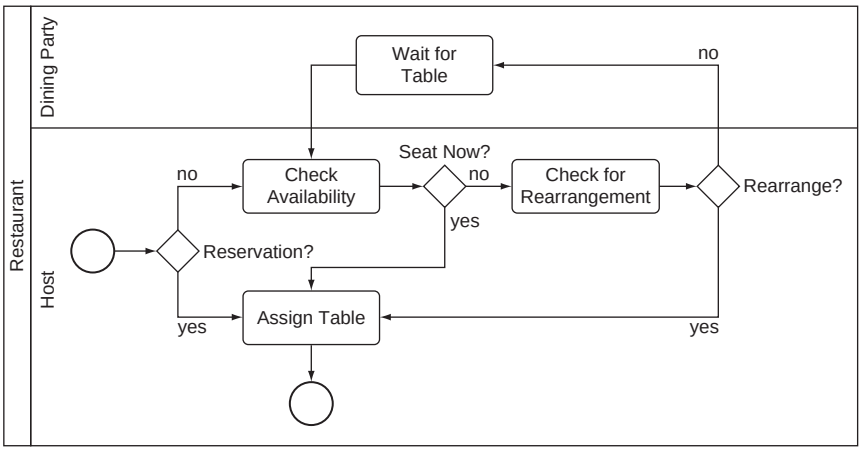

FIGURE 5.10 Seating flow

Parallel Gateways

The gateways in Figures 5.8, 5.9, and 5.10 are *exclusive gateways*. With an exclusive gateway, either one sequence flow is taken or the other is taken. But not every gateway is an exclusive gateway. A *parallel gateway* starts parallel work—two (or more) sequence flows that then progress at the same time, perhaps to be later joined back together by another parallel gateway.

Consider the process shown in Figure 5.11, with detail on the preparation of appetizers, entrees, and desserts. In Figure 5.11, the chef prepares the appetizers and the entrees at the same time. The appetizers can be prepared quickly and are served to the customers when they are ready. The parallel gateway **Split Order** splits the work into two parallel flows, one traveling the upper sequence flow to **Prepare Entrees**, and one traveling the lower sequence flow to **Prepare Appetizers**. After the appetizers and entrees are served, the sequence flows arrive at the other (unnamed) parallel gateway. At this point they are combined back together, and the subsequent activity **Serve Desserts** is performed only once.

Inclusive Gateways

Figure 5.11 assumes that each party orders both appetizers and entrees. But what happens if a dining party orders just entrees, without any appetizers? Figure 5.11 is not a good model of that situation because the parallel gateway mandates the use of both sequence flows. Figure 5.11 says that both entrees and appetizers are prepared.

Instead of using a parallel gateway, this situation can be modeled with an *inclusive gateway*, as shown in Figure 5.12. An inclusive gateway allows either

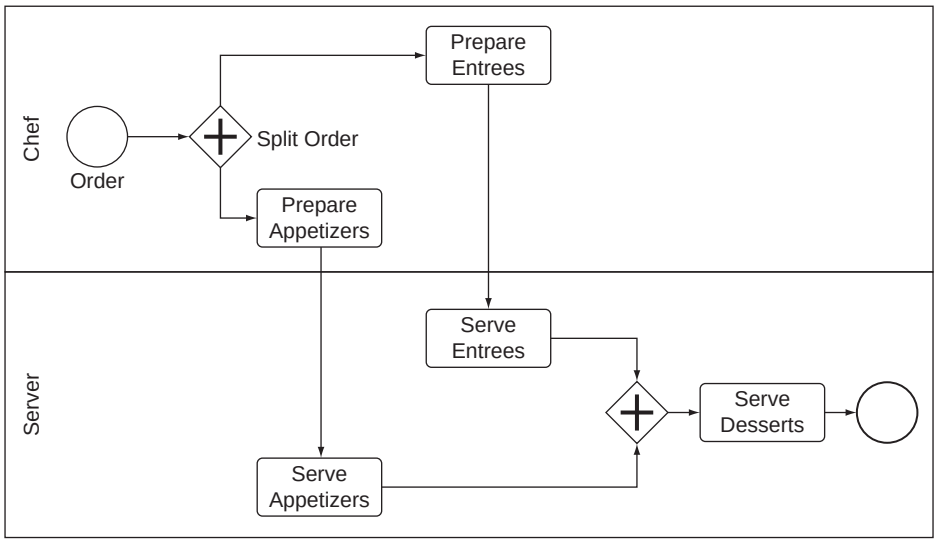

FIGURE 5.11 Using parallel gateways to prepare dinner

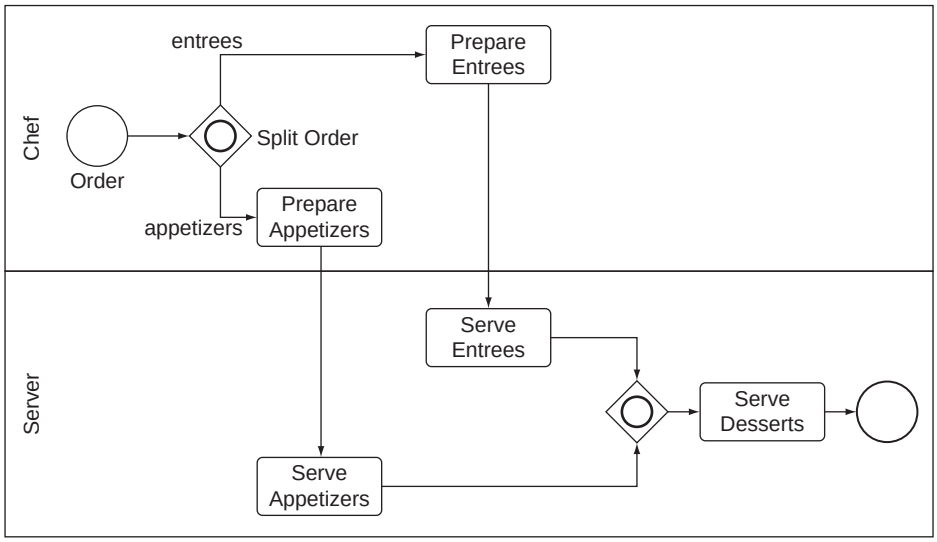

FIGURE 5.12 Using inclusive gateways to prepare dinner

outgoing sequence flow to be taken or both to be taken in parallel. A party can order just appetizers, just entrees, or both appetizers and entrees.

The behavior of the first gateway—**Split Order**—is a bit complex. When a work arrives at **Split Order**, sometimes it needs to travel the upper path, sometimes the lower path, and sometimes work needs to be split so that the two paths can occur in parallel. Chapter 11 describes how to simulate an inclusive gateway, providing percentages (for example) for how often the upper and lower paths are taken.

The behavior of the second inclusive gateway—the unlabeled gateway in Figure 5.12—depends on the behavior of the first, **Split Order**. For a party that orders only appetizers, when the appetizers are served, the second gateway passes the work through to the next activity, **Serve Desserts**, as though the gateway did not exist. Similarly, for a party that orders only entrees, the second gateway passes the work through to **Serve Desserts**. But for a party that orders both appetizers and entrees, the second gateway behaves like the unnamed parallel gateway in Figure 5.11. After the appetizers are served, the outgoing sequence flow waits at the gateway for the entrees to be served. Only when both the appetizers and the entrees have been served will the activity **Serve Desserts** be performed.

DEFAULT SEQUENCE FLOWS AND CONDITIONAL SEQUENCE FLOWS

One of the outgoing sequence flows from a gateway can be marked as a *default*. A default sequence flow is the one taken if there is no reason to take another sequence flow. Consider the process fragment on the left of Figure 5.13, showing what happens after drinks are offered to the diners. Either the diners order wine or they order mixed drinks or they order both. The default—indicated by the slash on the sequence flow—is that they order mixed drinks.

There is an alternative notation to using an inclusive gateway—an alternative way of showing the same process. The process fragment on the right of Figure 5.13 behaves the same as the fragment on the left, despite the different notation. There is no gateway on the right. Instead of a gateway, the sequence flow

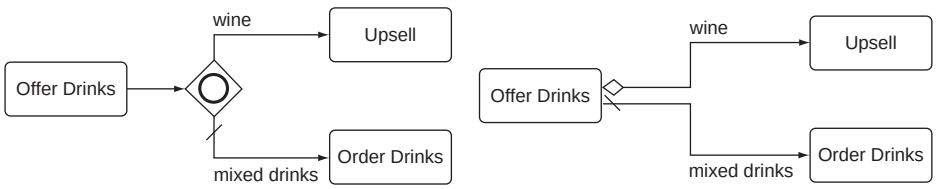

FIGURE 5.13 Identical process with a gateway and without

from **Offer Drinks** to **Upsell** is a *conditional sequence flow*. A conditional sequence flow is a sequence flow that includes a condition, a description of the situation under which it is permissible to take that sequence flow. The conditional sequence flow is depicted with a miniature diamond at its beginning.

SUBPROCESSES

Some activities are atomic; there is no more detail about the activity than its name, its description, and its attributes. Such activities are called *tasks*. The activities we have considered to this point are all tasks.

Often we are interested in understanding an activity in further detail. We want to break it down to a more detailed set of activities. A *subprocess* is an activity that has this extra detail, that can itself be described as a process. Figure 5.14 shows an end-to-end restaurant dining process. The party is greeted, is seated, orders dinner, dines, and pays. Each of the five activities is itself a subprocess, as shown by the + icon at the bottom of each activity.

Figure 5.15 shows one of the subprocesses—**Seat Party**—in detail. Figure 5.15 has the same activities, gateways, and sequence flow as the process fragment in Figure 5.10, but in Figure 5.15 the activities, gateways, and sequence flow are framed as a subprocess. Figure 5.15 has a start event and an end event. The start event in Figure 5.15 is not the start of the whole process. Instead it is only the start of **Seat Party**, the subprocess. Similarly, the end event in Figure 5.15 is only the end event for **Seat Party**.

How do Figures 5.14 and 5.15 work together? Figure 5.14 provides the context and launching points for the subprocesses, including **Seat Party**. After the dining party is greeted, they arrive at the subprocess **Seat Party**. The party then continues to the beginning of the **Seat Party** subprocess, the start event in Figure 5.15. The party moves within the subprocess, though the activities and gateways, until a table is assigned. When the party reaches the end event in Figure 5.15, the party returns to Figure 5.14, continuing to the next activity, **Take Order**.

Figure 5.15 shows the subprocess detail within one of the activities in Figure 5.14. When one diagram shows the subprocess of an activity from another diagram, the two diagrams are often referred to as the *lower-level process* and

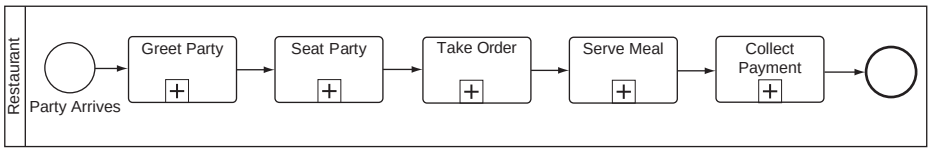

FIGURE 5.14 Five subprocesses

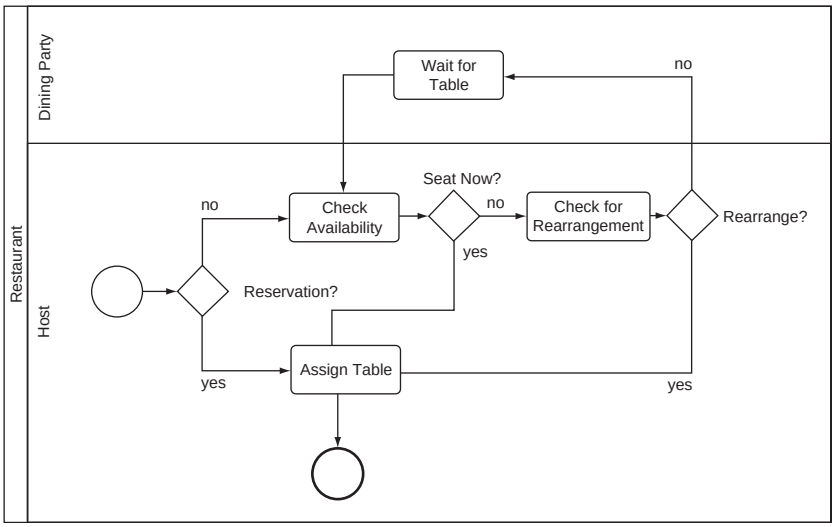

FIGURE 5.15 The subprocess within **Seat Party**

the *upper-level process*. The lower-level process—Figure 5.15—shows the detail of one of the activities in the upper-level process—Figure 5.14.

Figure 5.15 is an *independent subprocess*. The lower process is shown as a separate free-standing diagram. Some subprocesses are *embedded* instead of independent. The lower process details of the subprocess are shown in the context of the upper process, on the same diagram. Figure 5.16 shows **Greet Party** as an embedded subprocess. The **Greet Party** activity is large and contains three activities, joined by sequence flow. In many business process modeling tools, the embedded subprocess can be expanded by pressing the + icon. For example, in Figure 5.16, the user could press the + icon on **Seat Party**, causing the **Seat Party** activity to be expanded into an embedded subprocess, akin to **Greet Party**. The detail can subsequently be compressed back into the activity by pressing the − icon.

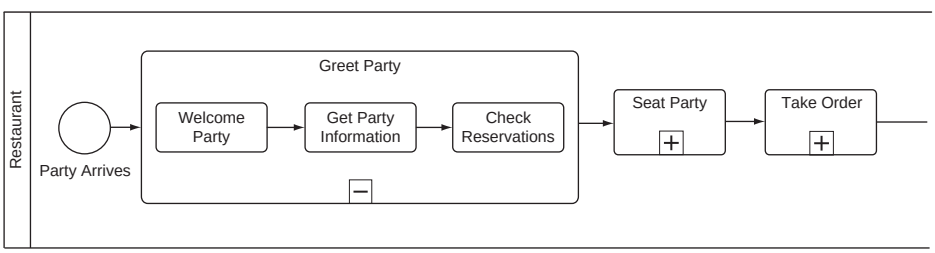

FIGURE 5.16 An embedded subprocess

The drawback of embedded processes is spacing: The diagram must be drawn with sufficient space to allow for the expansion of the subprocess. If **Seat Party** in Figure 5.14 were embedded, the whole process would have to be redrawn to provide space for **Seat Party** to be expanded within the lane. Embedded processes are impractical for modeling anything more complex than a few activities. It is also difficult to model an embedded subprocess within another embedded subprocess. In practice, most subprocesses are independent.

Figure 5.17 shows **Collect Payment**, another subprocess of Figure 5.14, originally shown as Figure 5.7. Unlike Figure 5.15, Figure 5.17 has no start event and no end event. A subprocess without a start event begins at an activity that has no incoming sequence flow. In Figure 5.17, there is one such activity, **Calculate Bill**, so the subprocess starts there. A subprocess without an end event finishes with activities that have no outgoing sequence flow. In Figure 5.17, **Release Table** is the sole activity without outgoing sequence flow. Omitting the start event and end event removes visual clutter while the process flow remains clear and understandable. Figure 5.18 shows the same **Collect Payment** subprocess with a start event and an end event. Figures 5.17 and 5.18 are equivalent; they behave the same.

In more complex subprocesses it is common to have two or more activities where the process ends. It is also possible to have two or more activities where the process starts. In such cases it is helpful to show all the start events and end events. Otherwise someone examining the process needs to locate all the activities that do not have an incoming flow and understand that they all start concurrently, and locate all the activities that do not have outgoing flows and understand that they are all possible end points.

Subprocesses provide several benefits. Subprocesses hide detail so that models can be divided up into chunks that are easier to manage and understand. A model with too many elements in a single diagram is difficult to understand.

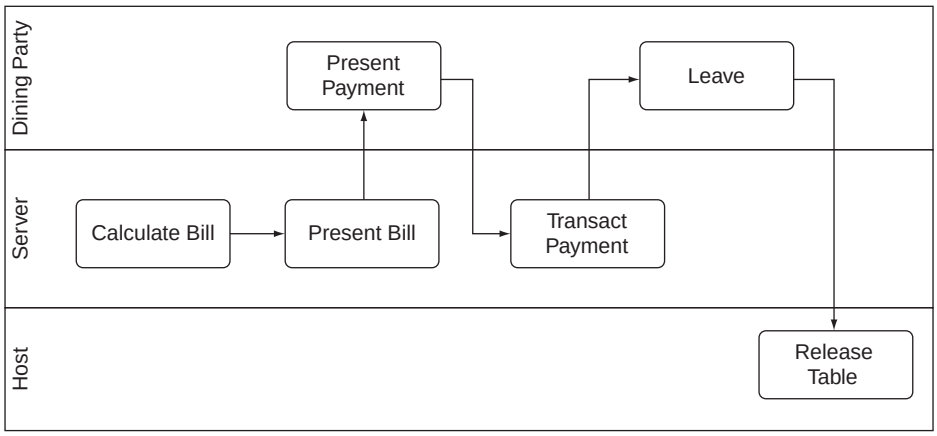

FIGURE 5.17 A subprocess without a start event or an end event

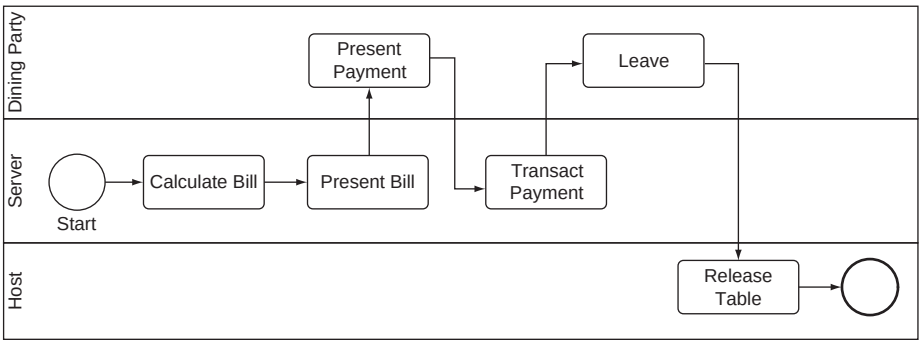

FIGURE 5.18 The same subprocess with a start event and an end event

Subprocesses are useful for modeling common process fragments that are used in several different locations. Rather than create the same process fragments several times, the same lower-level subprocess can be part of several different upper processes. Subprocesses reduce the modeling work that must be performed.

Subprocesses are also easier to maintain. If we add new activities, e.g., to implement a restaurant paging system (as described later in this chapter), we can add the activities within the **Seat Party** subprocess without affecting the rest of the overall restaurant process.

COMPENSATION AND OTHER CONDITIONS

A dining party might leave rather than wait for a table. Actually, a party could leave at any time—just after they arrive or when they learn of a wait, while waiting or after they see a menu. They might leave during the meal because their child is misbehaving or because someone in the party becomes sick.

There are many potential departures. It is possible to model all these potential departures with gateways. One could introduce a gateway after every activity, asking whether the dining party leaves now. The process model would then be full of gateways, all to model the situation of a dining party leaving early. Obviously, such a model is awkward.

Instead of using dozens of gateways, we model the potential for departure at any time with a single *exception flow*. An exception flow is a sequence flow triggered by an exception—an intermediate event that occurs sometime during the course of a subprocess. Figure 5.19 shows an exception flow. Attached to the boundary of the **Seat Party** activity is an intermediate event with a zig-zag icon. The intermediate event can occur at any time during the **Seat Party** activity. **Seat Party** has a subprocess shown in Figure 5.15, so the intermediate event can occur during any of the activities in the subprocess. The single intermediate event in Figure 5.19 eliminates the need to add several gateways to Figure 5.15.

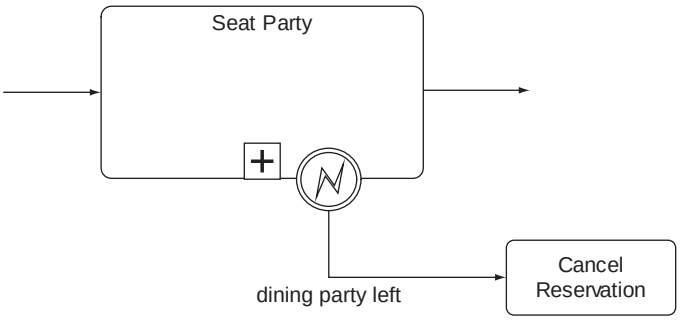

FIGURE 5.19 An exception and an exception flow

If the exception occurs in Figure 5.19 and the dining party leaves, their reservation needs to be cancelled. This cancellation is indicated by the activity **Cancel Reservation**, downstream of the intermediate event.

Business Transactions

Sometimes a business process includes a *business transaction*. A business transaction is a collection of activities that must either complete successfully or must be rolled back in their entirety, as though none of the activities had never been performed [Fowler 2003]. A business transaction can take hours, days, or even weeks to complete. Business transactions are common in business.

Let's consider an example. Some of the Mykonos restaurants have special event rooms that can be rented in their entirety for an evening. The reservations for these special event rooms are not performed using the usual dining reservations process at the individual restaurants. Instead Mykonos provides a centralized service for reserving special event rooms, so if a customer wants to reserve such a room in one restaurant and that room is already booked that evening, the centralized service can suggest reserving a special event room in another Mykonos restaurant across town.

Figure 5.20 shows the process for reserving a special event room. The normal process is quite simple. Mykonos takes a reservation for a room in the activity **Take Special Event Room Reservation**. Some customers want entertainment—a comic or musicians—for their event. Rather than allow customers to book their own entertainers, Mykonos prefers to handle those reservations as well, to ensure that only appropriate entertainment is performed at their restaurants. So the next step of the process is optionally booking entertainment in the activity **Reserve Entertainment**. Finally the customer is charged, two weeks before the event.

But the customer can change his mind before he is charged. If he cancels, Mykonos must cancel the reservation at the individual restaurant and cancel the

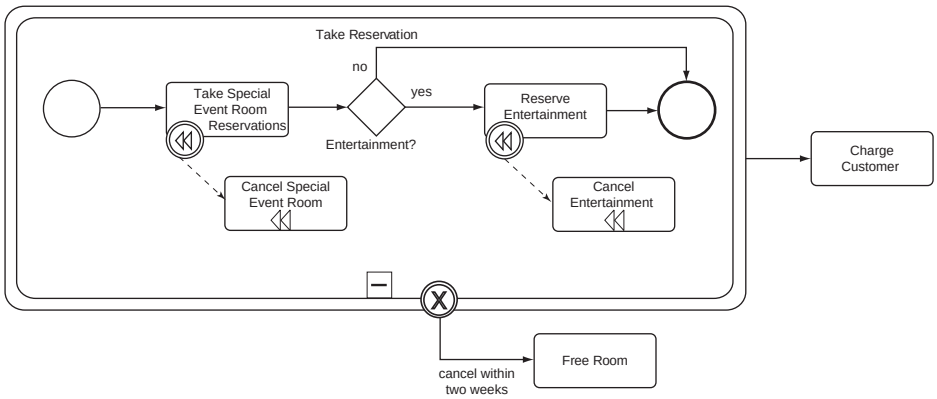

FIGURE 5.20 A transaction, a cancellation and some compensation

reservation for the entertainment. Figure 5.20 shows how this cancellation is modeled. **Take Special Event Room Reservation** and **Reserve Entertainment** are contained within the embedded subprocess **Take Reservation**. The subprocess is a transaction, as shown by the activity's double hull. The intermediate event with the cancel trigger—the outlined X—indicates that the transaction can be cancelled.

If **Take Reservation** is cancelled after the **Take Special Event Room Reservation** has occurred, the room will need to be cancelled. This cancellation happens in the activity **Cancel Special Event Room**. This activity is called a *compensation activity*, meaning that it compensates for activities that have already occurred, bringing the transaction back to the situation before anything happened. **Cancel Special Event Room** compensates for the activity **Take Special Event Room Reservation**. That compensation association between the two activities is depicted with a compensation trigger, the rewind marker on the boundary of **Take Special Event Room Reservation**. The corresponding compensation activity **Cancel Special Event Room** is also marked with a rewind marker. The two activities are connected with a dashed line, depicting the compensation association between them. Similarly, **Cancel Entertainment** compensates for the activity **Reserve Entertainment** if the customer cancels his reservation and entertainment has been reserved.

Only transactions can be cancelled; the cancel trigger is only allowed within a transaction activity. The intermediate event in Figure 5.19 is not a cancel trigger but is instead an exception trigger. So when a dining party leaves, no compensation activities will be performed.

The example shown in Figure 5.20 is rather simplistic—good for illustrating transactions and compensation but more simple than real-world situations.[1]

[1]Readers with a software engineering background will notice the difference between the business transaction in Figure 5.20 and database transactions. Database transactions are similar to business transactions but at a vastly different timescale—in milliseconds instead of days and weeks.

POOLS

Sometimes activities in a process must interact with another process. Supplies must be ordered via a separate procurement process. Taxes must be prepared and filed with national and local governments, each of which has its own processes for finding mistakes and violations. Leases are renewed for restaurants, involving negotiations with the real estate owners over price and terms.

We model multiple processes and the interactions between those processes using *pools*. A pool is a horizontal container for other process elements: activities, events, and gateways. A pool is a bit like a lane—as you recall, also a horizontal container for process elements—except that a pool can contain lanes, and usually does.

Consider an example in which a Mykonos procurement specialist orders kitchen equipment (e.g., a stove) from a distributor, shown in Figure 5.21. The process begins when the procurement specialist places an order for the equipment. The distributor's sales department receives the request and verifies with the warehouse that the item is in the inventory. Sales then calculates the price and creates an invoice that is sent to the restaurant. The procurement specialist receives the invoice and issues a deposit payment for the equipment. Once Sales receives the deposit, it approves the order and instructs the warehouse to ship the equipment. The process ends when the restaurant receives the equipment.

Like many of the restaurant example models in this book, the process shown in Figure 5.21 is simple. It is meant to illustrate, not to be complete. We did not

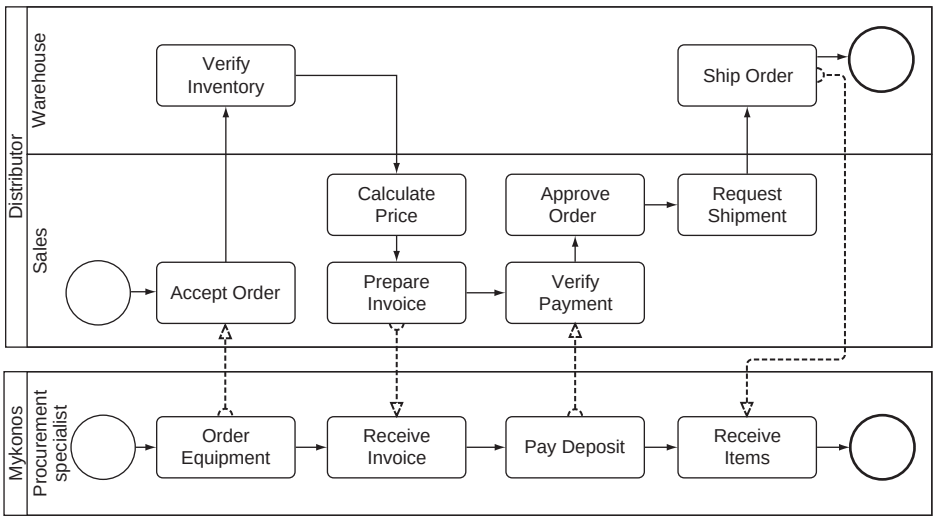

FIGURE 5.21 A process with two pools

include what happens if the warehouse is out of stock or if partial payment is received. Similarly, when the equipment is received it is inspected and if inspection reveals something wrong, it is returned. This process Mykonos employs is more complicated than the model shown in Figure 5.21.

Figure 5.21 has two pools: a pool at the top and a pool at the bottom. The pool at the top shows the distributor's process. That pool has two lanes: one for the warehouse and one for sales. The pool at the bottom shows Mykonos's activities and has a single lane for the procurement specialist.

You might be wondering why pools are necessary. Why not model the process of Figure 5.21 with three lanes and no pools, as shown in Figure 5.22? What's the difference between the two models? The difference between Figure 5.21 and Figure 5.22 is profound. The process shown in Figure 5.22 is managed as a single process. Some organization—perhaps Mykonos—has the responsibility for the whole end-to-end process. If something goes wrong, the responsible organization (i.e., Mykonos) is charged with fixing the problem. So, if sales cannot calculate a price for some reason and the order sits at the **Calculate Price** activity, Mykonos has the authority to solve the problem, perhaps to help them calculate the price and then to move the order along to the next activity.

But in fact Mykonos does not have that authority over the activities in the sales lane. The sales participant is an employee of the distributor, a different company, over which Mykonos has no control. And the distributor has no authority over the activities in the procurement specialist swimlane. Figure 5.22 is a poor model of the equipment procurement process because it implies that some organization has authority over the whole end-to-end process. That single organization does

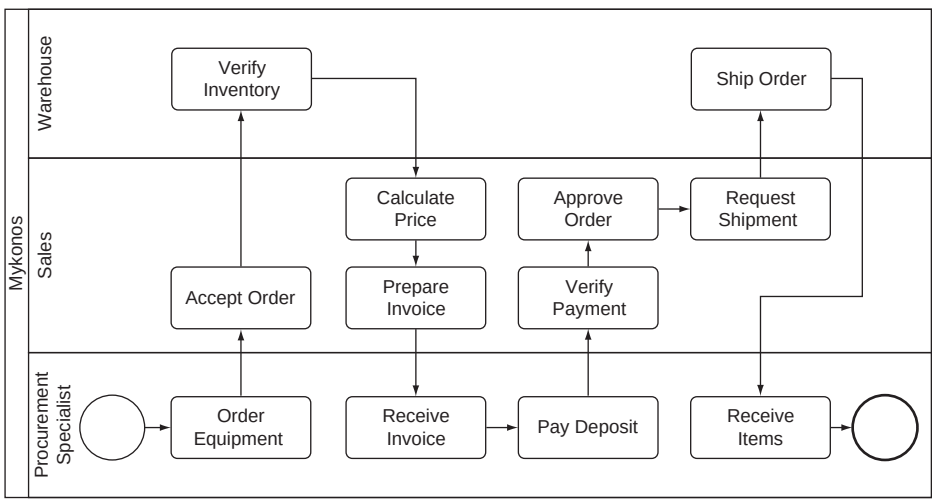

FIGURE 5.22 Three lanes instead of two pools

not exist. Figure 5.21 is a much better model. Mykonos has responsibility for the activities the bottom pool, and the distributor has responsibility for the activities in pool at the top. No single organization manages procurement as a single process. Instead it is managed by two different organizations, as two processes that interact.

The interaction between the pools occurs in *message flows*, shown as the dashed lines in Figure 5.21. For example, there is a message flow between the activities **Order Equipment** and **Accept Order**. A message flow is different from a sequence flow. A message flow is used to connect activities (or events) that are in different pools, modeling a flow of messages between the two activities. For example, as part of the **Order Equipment** activity, the procurement specialist sends a fax to the distributor. That fax is read and interpreted in **Accept Order**. The sending of the fax is a message and is modeled with a message flow. Message flows also model other means of messaging, in addition to faxes: emails, Web service invocations, telephone calls, even in-person visits. All are messages; all are modeled with message flows.

Message flows are only used between pools. Two activities in the same pool are never connected by a message flow, because both activities have access to the same information. All activities in the same pool are assumed to have access to whatever information they need, through IT systems, paper, or verbal communication. Similarly, sequence flows are never used to connect activities in different pools. As you recall, a sequence flow between one activity and another means that the second activity happens after the first is completed. A single organization manages that process, even when the two activities are in separate lanes. In Figure 5.21, **Verify Inventory** and **Calculate Price** are connected by a sequence flow, so **Calculate Price** happens after **Verify Inventory** is complete. The distributor organization ensures that **Calculate Price** takes place at the right time. But as we have discussed, there is no single organization with authority across pools. Sequence flows are inappropriate between pools because there is no one to ensure that the sequencing occurs.

In Figure 5.21 both pools are shown as *white boxes*. A white-box pool is one in which all the activities are modeled. In Figure 5.21 we can see all the activities in the distributor pool and all the activities in the Mykonos pool. Everything is visible.

White-box modeling is often impractical. Often you do not know how your business partners do what they do. You know only how your own process interacts with their process. And it is often not so important to understand the internal details of your business partners' processes. For these situations, *black-box* modeling is more appropriate. In a black-box business process model, the internal details of the external organizations are not shown. Figure 5.23 shows the same process as Figure 5.21 but with the distributor pool as a black box.

Figure 5.23 has the same messages as Figure 5.21, connecting to the same activities in the Mykonos pool but connecting to the distributor pool, not to activities but to the pool as a whole. For example, in Figure 5.21 there is a message flow from **Order Equipment** to **Accept Order**. The same message flow is shown in Figure 5.23, but the message flow is from **Order Equipment** to the black-box

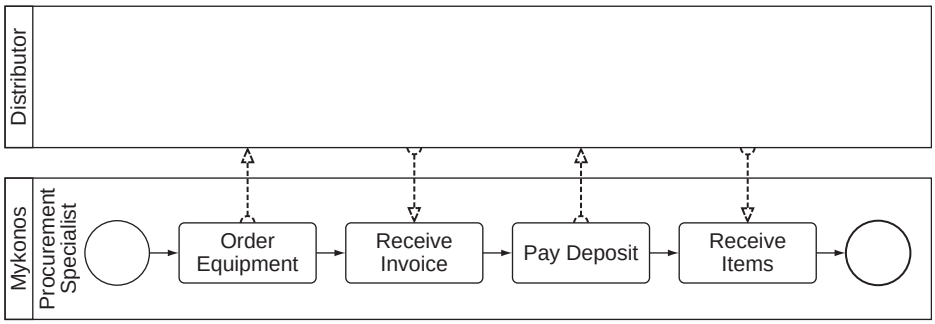

FIGURE 5.23 A black-box pool

distributor pool. In Figure 5.23 we can see exactly how Mykonos is interacting with the distributor, but we cannot see inside the distributor's process.

How do you decide whether a business process should be modeled using multiple pools interacting with message flows or as a single pool using sequence flow? The key question to answer is: Who manages the process? If the whole process is managed by a single organization, a single pool is appropriate. If multiple organizations manage their own processes that then interact, multiple pools are important.

As we write this chapter, there is some difference of opinion in the business modeling community about when pools are appropriate. Some modelers think participants who are part of different organizations must always be in different pools. So these modelers consider the process shown in Figure 5.15 to be invalid, since the dining party is certainly not part of the restaurant organization. These modelers would draw Figure 5.15 in two pools, with a message flow between **Wait For Table** and **Check Availability** modeling the verbal communication that occurs when an impatient dining party asks whether a table is ready now.

We think this approach is overly restrictive, and leads to overlay complex models. We believe it is appropriate to include an external participant within a pool as long as a single organization is managing the whole process. In Figure 5.15, **Dining Party** is included in the **Restaurant** pool because the restaurant is managing the whole process. As always, you are free to use either approach, as long as you are consistent.

AS-IS AND TO-BE PROCESSES

A model of today's business is often called an *as-is model*, to contrast with a model of a desired future situation, a *to-be model*. The terms "as-is" and "to-be" work for all business modeling disciplines. One can have an as-is process model and a to-be process model, showing activities, gateways, and sequence flow for today and for the planned future. Similarly, one can have an as-is organization model and a to-be organization model, showing today and tomorrow for the organization. One can even have as-is and to-be motivation models if the strategy is to be changed.

It is common to have a single as-is process model and several different to-be process models. Several alternatives are evaluated to decide which is the best business processes for the future. Depending on the evaluation goals, we could decide on a to-be process that takes the least amount of time to complete the work. Alternatively, we might decide on the to-be process that best improves customer quality.

It is also possible—albeit less common—to have several as-is processes. For example, Mykonos restaurants perform customer reservations differently. Some restaurants call the customer on the day of the reservation to confirm and some do not. Some restaurants only take reservations two weeks in advance, some limit reservations to one month forward, and Portia will take a reservations one year in advance. When Mykonos implements a reservation application across all its restaurants, they will standardize on a single reservation process, modeling all these alternative as-is processes to understand what is involved in the standardization.

Let's look at an example of an as-is process. Figure 5.24 shows a single as-is process: the subprocess for seating customers. The as-is process is purely manual; No technology supports the host in her job of seating the customer. A dining party waits until a table is available, and they either check periodically with the host or just wait until the host informs them that a table is available.

Studying the as-is process provides an understanding of how work is performed today as well as some insight into problems. For example, when a table is not available, the dining party waits in the **Wait for Table** activity. For some customers, this is an anxious wait because they are unsure whether they have been forgotten.

At some family-oriented Mykonos restaurants, pagers are provided to waiting diners. Pagers make the waiting less anxious for some diners, because they do not have to periodically ask the host if they have been forgotten. Diners have a feeling of assurance that they are queued in a system, and they can relax and enjoy waiting for their pager to buzz.

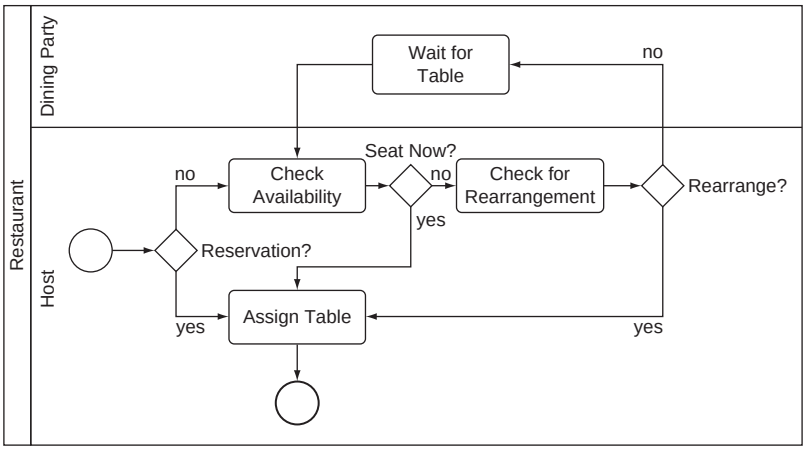

FIGURE 5.24 Seating subprocess

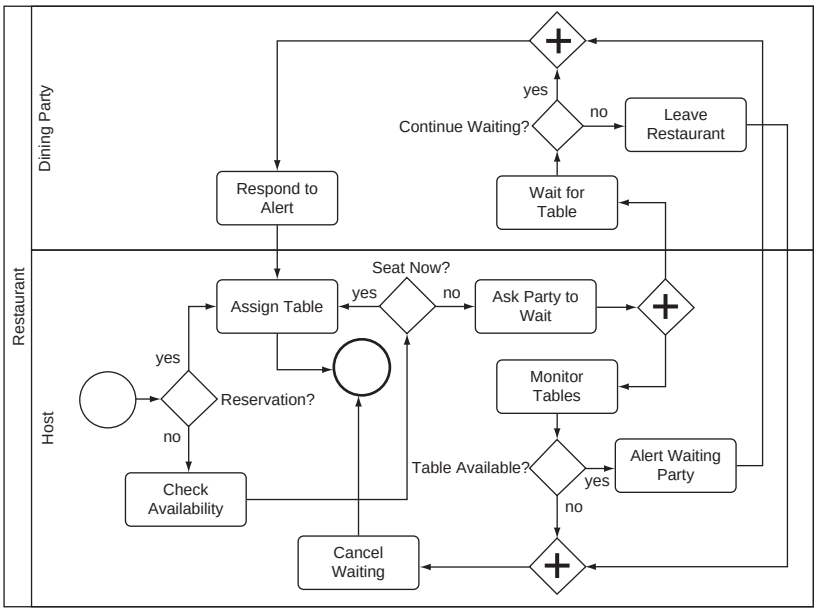

FIGURE 5.25 The to-be process with a pager

Figure 5.25 shows a revised **Seat Party** subprocess, incorporating activities that use a paging system. Figure 5.25 is a to-be process, one potential future. When a dining party is asked to wait in **Ask Party to Wait**, they are given a pager. At any point the party might choose to leave. If they leave they return the pager. If they stay, when the pager notifies them that a table is available, they return the pager, the host assigns the table and the diners are seated.

By comparing the to-be process of Figure 5.25 with the as-is process of Figure 5.24, we can see which activities are performed differently and which new activities are needed.

BUSINESS MOTIVATIONS AND PROCESSES

In Chapter 3 we described goals, courses of action, strategies, and other elements of business motivation models. As you will recall, a goal is something an organization is trying to achieve, a course of action is something an organization does to achieve a desired result, and a strategy is a broad, lasting course of action. Figure 5.26 shows the restaurant **Portia**, an organization unit; the goal **Improve Customer Experience** that Portia defines; and the strategy **Use Pagers** that Portia establishes to channel efforts toward the goal.

Organizations are responsible for business processes. An organization is responsible for a business process if the organization is charged with the successful completion of the process and with fixing any issues that arise along the way. In Figure 5.26, **Portia** is responsible for the process **Seat Party with Pagers**, the business process detailed in Figure 5.25.

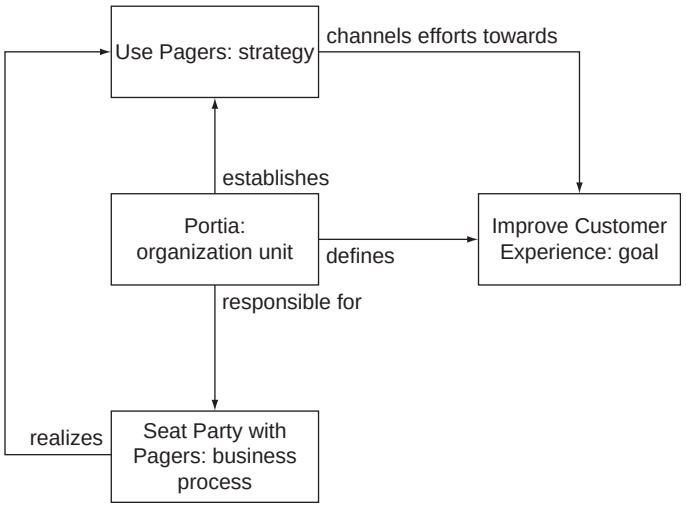

FIGURE 5.26 Business motivation and business processes

Business processes *realize*[2] courses of action. Realizing a course of action means that the business process implements the course of action—that the activities, gateways, and sequence flow in the business process achieve the course of action. Figure 5.26 shows that the **Seat Party with Pagers** business process **realizes** the strategy **Use Pagers**.

An individual activity can also realize a course of action. Figure 5.27 shows the individual activities within the **Seat Party with Pagers** business process that realize the strategy **Use Pagers**. Although the process as a whole realizes the strategy, only some of the activities within the process realize it.

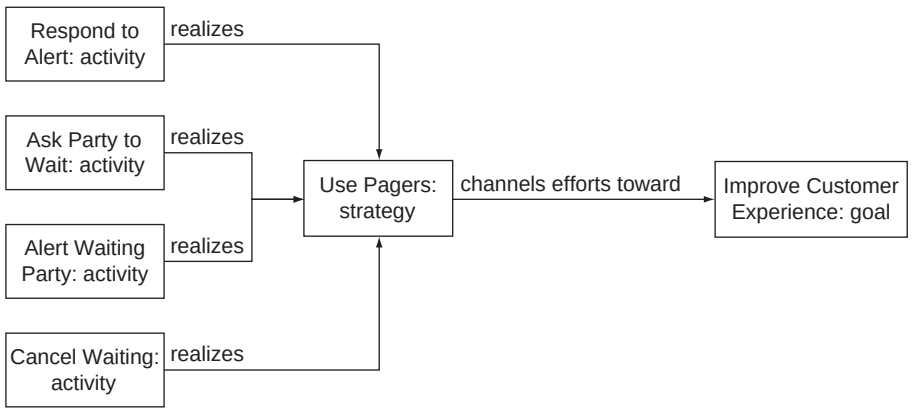

FIGURE 5.27 Business motivation and activities

[2]As you will recall from Chapter 3, BMM is the OMG standard for business motivation models. The **realizes** association between a business process and a course of action is introduced in BMM.

BUSINESS PROCESSES, ORGANIZATIONS, AND INTERACTIONS

In Chapter 4 we described organizations, roles, and other elements of business organization models. As you will recall, an organization is a collection of people who work together toward a common goal. A role is the responsibility a person assumes when he holds a position in an organization. Business process models also show organizations and roles. Each lane in a business process model is labeled with the participant who performs the activity. The participant is either an organization or a role. Similarly, each pool is labeled with the participant who manages that process. That participant is also either an organization or a role.

Figure 5.28 shows the relationship between a business organization model and a business process model. At the bottom of Figure 5.28 is the business process model of the Mykonos procurement of kitchen equipment, originally shown as Figure 5.21. At the top of Figure 5.28 is a business organization model that describes the relationship between Mykonos and the distributor, the same distributor role shown in the process model at the bottom. Between the top and bottom are arrows showing which model element at the top is the same as which label at the bottom.

The organization Mykonos is shown at the top of Figure 5.28. The **Operations** organization unit is within **Mykonos** because it is part of the larger restaurant company. Within **Operations** are three organizations: **Human Resources, Finance**, and **Procurement**. The **Procurement Specialist** role is part of the **Procurement** organization. There is an arrow from the **Procurement Specialist** role at the top of Figure 5.28 to the lane label of the lower pool at the bottom. The **Procurement Specialist** role is the participant of that lane; all the activities in that lane are performed by someone who takes that role. There is also an arrow from **Mykonos** to the pool label of the same lower pool. Mykonos is responsible for the process of the lower pool.

Figure 5.28 also shows the **Distributor** role. **Distributor** has two organizations within: **Sales** and **Warehouse**. Both of those organizations are participants of lanes in the upper pool, as shown by the arrows. The distributor is responsible for the upper pool process as a whole.

Not every organization or role in a business organization model corresponds to a participant in a business process model. Instead only the relevant organizations and the relevant roles are participants. For example, **Human Resources** is an organization within Mykonos but not a participant of any lane or pool in the equipment procurement process, since no one from Human Resources does any of the work in this process.

There are two interactions in the business organization model of Figure 5.28. One interaction is **order**, between **Procurement Specialist** and the **Sales** organization. The other interaction is **kitchen equipment** delivered by the **Warehouse** organization to the **Procurement Specialist**. As described in Chapter 4, the direction of an interaction is from the organization (or role) delivering the value to the organization (or role) receiving the value.

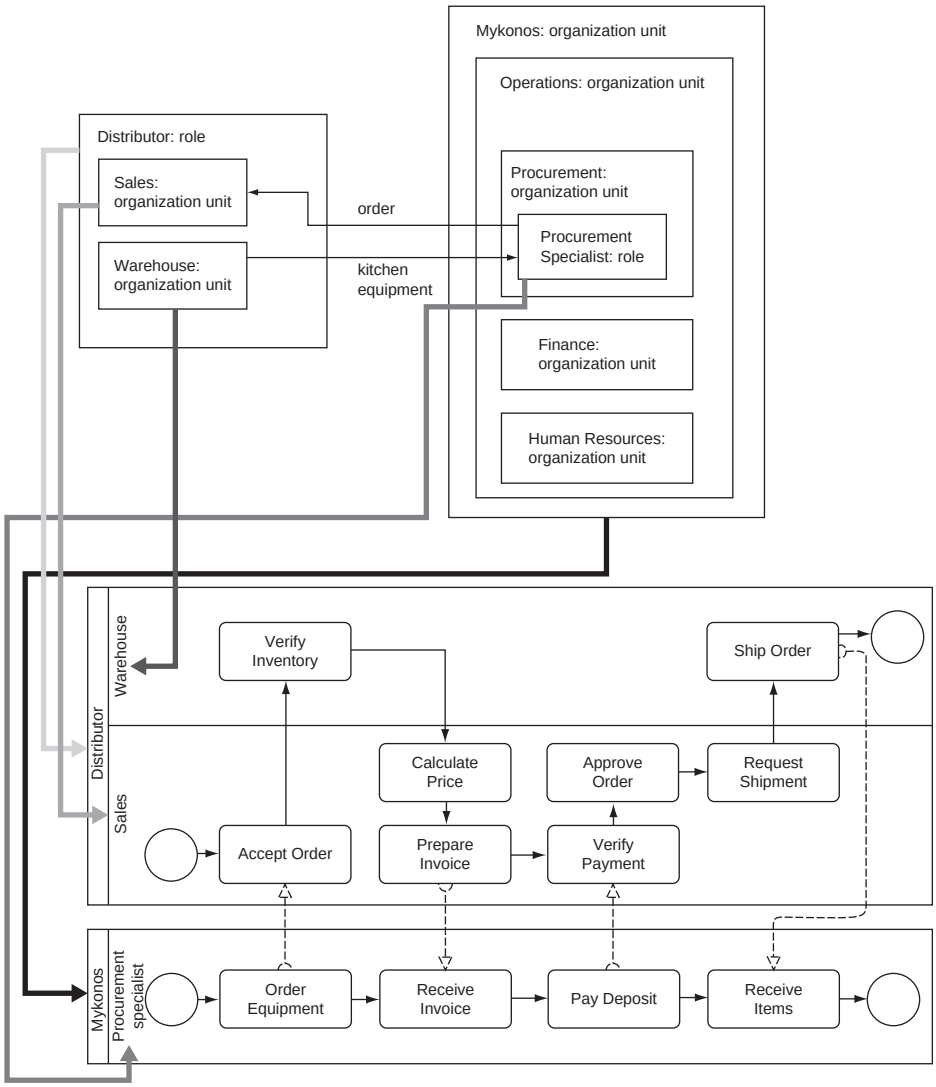

FIGURE 5.28 Business organizations and business processes

An interaction between two organizations in a business organization model can correspond to a business process in a business process model, with the business process detailing the way the two organizations actually work together. The business process at the bottom of Figure 5.28 details the **order** interaction between the distributor's sales organization and Mykonos's procurement specialist, showing

all the activities that result in the order being placed. The same process also details the **kitchen equipment** interaction between **Warehouse** and **Procurement Specialist**.

THE BPMN STANDARD

The Business Process Model Notation—commonly known by its acronym, BPMN—was originally published in 2004 by the organization Business Process Management Initiative. BPMN is a graphical modeling notation for business processes that is independent of a specific implementation environment. BPMN was officially adopted as an OMG specification in 2006, and updated in 2008 [OMG 2008b].

BPMN is a good standard with modeling constructs for all the common business process modeling needs and for many less common needs as well. BPMN is a successful standard, becoming widely adopted in the business process modeling community. The business process models in this chapter, and through this book, are shown in BPMN. As with other business modeling standards, we make no attempt at completeness in our description. We use only some of the elements of BPMN—the elements we find most useful in our everyday business process modeling.

Prior to the emergence of BPMN, modeling practitioners created business process models using a great variety of notations. There are several other standards and many nonstandard proprietary methods for the visualization of business processes. In our experience, each of these standards and proprietary methods has shortcomings that limit its usefulness and in practice have limited its use.

IDEF was one of the standard notations used for modeling business processes. IDEF was developed by the US Department of Defense (DoD) in the 1980s, primarily for the modeling of information and software but also widely applied to business processes. IDEF includes 15 modeling methods—IDEF0 through IDEF14—to support a variety of purposes. IDEF0 and IDEF3 have been used for business process modeling.

IDEF0 shows business process activities but does not show their sequence or timing. In practice, many readers of IDEF0 models found them difficult to understand because it was not apparent what activities occurred when. IDEF3 was later developed to show sequence flow among activities. Today IDEF is still used by DoD and its contractor community but is not popular in other organizations. Businesspeople find IDEF difficult to understand.

UML is another standard notation that has been used for modeling business processes. As described in Chapter 1, UML is a software modeling language. One of the diagrams of UML—activity diagrams—has been used for modeling business processes [Eriksson 2000]. UML activity diagrams are functionally similar to BPMN diagrams, and can be used to model business processes. But UML looks quite different from BPMN. BPMN was designed to be understood by businesspeople—people

without training in software development. By contrast, UML users are technical people: software developers, software architects, and system architects. UML diagrams look technical, and in practice they are much harder for businesspeople to understand than BPMN diagrams. Furthermore, BPMN has a richer set of model constructs for business process modeling—constructs including pools, compensation, and timer start events.

BPMN is designed for business modelers and provides those modelers with a rich but simple notation to create business process models. As of this writing the tool support for BPMN is still evolving. Though many tools claim to implement BPMN, some have implemented only portions of the standard—the portions that are most similar to the proprietary modeling techniques they already support. Business process models cannot yet exchange models; a model created in one tool cannot yet be read into another tool. At the time of this writing, the BPMN standard is being refined within the OMG. We expect future versions of BPMN will provide guidance for tool interoperability, as well as provide additional modeling features.

As more tools support BPMN and provide for conversions between their own formats and BPMN, we expect BPMN to be an excellent choice for avoiding dependence on a single vendor's proprietary modeling method or tool. And today BPMN already provides a rich set of modeling constructs, providing all the constructs available in proprietary tools and several that are unique, including business transactions and compensation. We find no reason to use any other business process modeling notation.

OMG has also adopted a second standard for business process modeling: Business Process Definition Metamodel (BPDM), adopted in 2007. BPDM is not an alternative to BPMN. Instead it complements BPMN. Whereas BPMN specifies the notation, the visual look of a business process model, BPDM supports the representation of process models independently of any notation. BPDM provides a serialization capability for BPMN, so a model can be saved to a file in a standard way. BPDM also deals with the synchronization and execution order of a business process and describes the way participating organizations in a process define their agreements and interactions so that they can collaborate.

BPMN and BPDM address different audiences. Though BPMN addresses business modelers, BPDM addresses tool vendors that need to support import and export capability of the models in their tools.

Because they are complementary standards, OMG plans to merge BPMN and BPDM into a single standard, with the merging planned for BPMN version 2. Today, even when a tool supports portability to another tool, the model elements can be moved, but their visual layout is lost. A modeler must recreate the model diagram, repositioning each model element. Version 2 of the combined standard will include support for diagram layout interchange so that model layout is also preserved.

Case Study

The United States shares a 2,000-mile border with Mexico. That 2,000 miles is the most frequently crossed international border in the world, with 250 million legal crossings every year. In addition to the many legal crossings, there are also some illegal ones. Every year an unknown number of people—perhaps as many as a million, perhaps more—cross the border from south to north, from Mexico to the US. Most of the people crossing are economic migrants, attempting to create better lives for themselves by working in the United States. Some of the people are smugglers, carrying illegal drugs and other contraband. And among the large numbers of economic migrants and smugglers, it is believed that terrorists, planning to inflict religiously-inspired violence on Americans, might also try to cross the southwest border.

US Customs and Border Protection (CBP) is responsible for preventing illegal border crossings. At the time of this writing, CBP has only partial success in catching people who cross the border illegally. Historically, most people who attempted to cross ultimately succeeded (even if it took several attempts). In 2006, CBP sought technology solutions for improving their performance, for catching significantly more illegal crossers, enough to deter crossers from attempting to cross. They were looking for a private sector partner to create something—a wall, a fence, electronics, or something else—that would help them spot and apprehend all the illegal border crossers.

CBP created a competition among prospective private sector partners to determine who could propose the best solution to their challenges. Five companies responded. Four recommended mixes of sophisticated technologies, including watching the border with blimps or unmanned aerial vehicles. Our approach was different. Instead of merely providing a technology-centered approach, we also focused on the business processes. We learned as much as we also could about the everyday life of border patrol agents, the law enforcement officers who patrol the border. Our approach involved five steps:

1. Understand the government agencies involved, how each works, and how each interacts with other agencies.
2. Capture as-is business process models, to understand how work is being done today and the challenges with the existing process.
3. Create alternative to-be business process models, to investigate how work will be performed after the deployment of various solutions.
4. Simulate the alternative to-be process models, to understand which are most effective.
5. Map the proposed technology solutions to the to-be business processes, to understand how the technologies will be used.

The business processes we modeled encompass the entire cycle of handling illegal border crossers—everything from the detection of an illegal crosser to that person's ultimate removal from the United States. The entire cycle includes detection, identification and classification, response, apprehension, detention, and removal. Detection is the initial spotting of a group of illegal crossers. A group might be detected by sight, or technology such as radar and cameras can be used. Once a group is detected, the illegal crossers are identified and classified. Are the crossers a family with children carrying luggage, economic migrants who are likely nonviolent? Or are they four fast-moving individuals carrying suspicious backpacks, smugglers who are likely to be dangerous? Response involves dispatching the appropriate number of border patrol agents and dispatching them to the right location. Apprehension involves arresting the border crossers and transporting them to a border patrol station. After they arrive at the station, the border crossers are fingerprinted to identify whether anyone is wanted for another criminal activity. The border crossers are interviewed to determine who is a Mexican citizen and who is a citizen of another country. Mexican citizens are returned to Mexico in a bus, but anyone from another country cannot be returned to Mexico. They are instead deported to their country using a far longer process, based on international laws and agreements.

CBP was interested in solutions for detection, identification and classification, and response. But we went much further in our investigation, looking at the downstream processes of apprehension, detention, and removal. These downstream processes were important for providing us with a complete understanding of the situation and the challenges facing CBP.

We worked with border patrol subject matter experts—retired border patrol agents who provided us with detail about the activities they performed and the challenges of their day-to-day work. Working with subject matter experts provided insight into how they performed their work. For example, we learned that each geographical location along the border is different, with different terrain, different weather, and different socioeconomic situations. One technology solution cannot fit all the locations.

We explored the suitability of new technology with the subject matter experts, looking at ways different technology solutions would affect the agents' work. Would a particular new technology help or hurt them? Agents already carry a good deal of equipment: radio, multiple cellphones, flashlights, and weapons. Adding more equipment weighs them down. It is better to replace equipment, to make the whole load easier to carry. Also, most illegal crossing happens at night, under the cover of darkness. If an agent uses an electronic device with a lighted screen, the device will illuminate the face of the agent, exposing his location to all the illegal border crossers, and making it easier for

Continued

Case Study—continued

them to avoid him. The lighted screen would even make him an easy target for those border crossers who are violent.

Figure 5.29 focuses on the transportation activities, showing the embedded subprocess **Transport Crossers**. Transportation historically has been performed by border patrol agents but does not require their specialized tracking and apprehension skills. For a small group of illegal crossers, the agent drives the group to the border station. But if the group is too large to fit in his truck, he must wait with the apprehended border crossers for another agent to bring a van or a bus. While the agent is waiting and while he is driving, he cannot intercept any more border crossers.

We explored several alternatives to the current process. The best alternative was a combination of technology improvements, resourcing improvements, and process improvements. One improvement was a set of technologies to keep the agents in the field in constant communication with the stations. Other improvements included the ability to track positions of agents, track incidents they are handling, and track assets such as transport vehicles. This allows the dispatcher to determine the best and most efficient pick-up point, minimizing the wait and transportation time.

Resource combinations were also explored. What is the best number of vehicles to handle the expected number of crossers and minimize the overall wait times? Who should drive the vehicles so that the agents can spend their time catching illegal crossers?

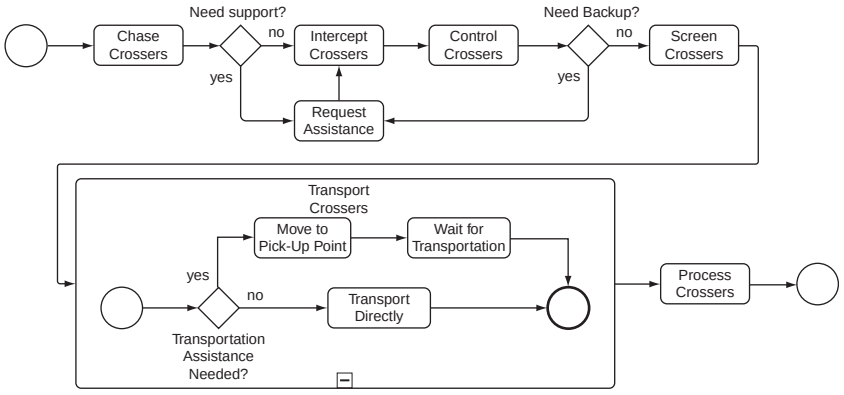

FIGURE 5.29 Transport Crossers subprocess

We used business process simulations to analyze the process, resulting in proposal recommendations. These recommendations involved the outsourcing of transportation, the number of buses, and technologies to better support the agent.

In September 2006 CBP awarded the contract to our team. Our proposal was recognized as having the best approach. In particular we were recognized for our deep understanding of CBP's business and needs.

Business process models are useful for understanding how activities are accomplished today but also for creating good approaches for how they should be accomplished in the future. They are useful for conveying and communicating an approach and for showcasing an understanding of the issues and solutions.

A business process model describes the work that is being performed, the order in which work is performed, and who performs the work. A business process model includes a rich set of modeling elements—activities, gateways, events, and flows— that describe the complexity of a process.

A business process model is about how work is accomplished. In Chapter 6, we examine the rules that govern a business process—the business guidelines that restrict what should be done when performing the business process.