# CHAPTER 6: BEHAVIORAL MODELING

Since Alec, Margaret, and the team have now completed rough functional and structural models for their evolving Web-based solution, they have decided that it was time to move on and begin to create the behavioral models. Alec understood that in some ways, the behavioral models allow them to complete their understanding of the problem. In this installment of the CD Selections case, the team creates sequence diagrams, communication diagrams, behavioral state machines, and a CRUDE matrix. As in the previous installments, we see how the team goes about creating, verifying, and validating the behavioral models of the Web-based system they hope to implement. As in Chapter 5, you should realize that the team created behavioral models for all of the use cases and classes in the evolving system description. However, in the sections that follow, we only see the models associated with the Place Order use case and the Order class. The sections are organized in the same manner as the chapter: sequence diagrams, communication diagrams, behavioral state machines, and CRUDE matrix.

## Sequence Diagrams

To begin with Alec decided that the team should follow the guidelines for creating sequence diagrams listed in Figure 6-4. Next, Alec decided that the team should follow six steps to create a sequence diagram described in the textbook. Consequently, Alec first needed to determine the context of the sequence diagram. He decided to use a scenario[4] from the Place Order use case that was created with the business process and functional modeling installment in Chapter 4 and illustrated in Figure 4-G. (Refer to the original use case for the details.) Figure 6-A lists the Normal Flow of Events that contains the scenario that this sequence diagram describes.

The second step was to identify the actors and objects that participated in the scenario being modeled. The classes associated with the Place Order use case are shown in Figure 5-D. For example, the classes used for the Place Order use case include Customer, CD, Marketing Information, Credit Card Center, Order, Order Item, Vendor, Search Request, CD List, Review, Artist Information, Sample Clip, Person, and Organization.

During the role playing of the CRC Cards, Anne, one of the analysts assigned to the CD Selections Internet System Development Team, asked whether a Shopping Cart class should be included. She stated that every Internet sales site she had been to had a shopping cart that was used to build an order. However, the instance of the Shopping Cart class only existed until either an order was placed or the shopping cart was emptied. Based on this obvious oversight, both the Place Order use case and the class diagram will have to be modified. Brian, another analyst, pointed out that the CDs themselves were

Normal Flow of Events:

1. Customer executes the Search/Browse CDs use case.
2. The System provides the Customer a list of recommended CDs.
3. The Customer chooses one of the CDs to find out additional information.
4. The System provides the Customer with basic information and reviews on the CD.
5. The Customer iterates over 3 through 4 until done shopping.
6. The Customer executes the Checkout use case.
7. The Customer leaves the Web site.

FIGURE 6-A
Normal Flow of Events of the Places Order Use Case

[4]Remember, as stated previously, a scenario is a single executable path through a use case.

going to have to be associated with some type of searchable storage. Otherwise, it would be impossible to find the CD in which the customer was interested. However, Alec decided that the CD List class would suffice for both the searchable storage and a temporary instance that would be created as a result of a search request. Alec pointed out to the team that this process was fairly typical in object-oriented development. The more the development team learned about the requirements, the more the models (functional, structural, and behavioral) would evolve. Alec reminded them that the important thing to remember was that an object-oriented development process was incremental and it iterated over all of the models. As such, as they understood the problem better, the team would most likely have to make changes to the functional and structural models already completed.

Based on the team's current understanding of the Place Order use case, they decided that instances of the Search Request, CD List, CD, Marketing Material, Customer, Review, Artist Information, Sample Clip, and Shopping Cart classes would be required to describe this scenario. Furthermore, they realized that the Customer actor interacted with the scenario. To complete this step, the team laid out the objects on the sequence diagram by drawing them, from left to right, across the diagram.

The third step was to set the lifeline for each object. To do this, they drew a vertical dotted line below each of the objects (aSR, aCDL, CDs, aCD, MI, aR, AI, SC, and anOrder) and the actor (aCustomer). They placed an X at the bottom of the lifelines for aCDL and aSC since they "go away" at the end of this process.

The fourth step was to add the messages to the diagram. By examining the steps in Figure 6-A, the team was able to determine the way in which the messages should be added to the diagram. Figure 6-B shows the diagram they created. Notice how they did not include messages back to Customer in response to "create SR" and "add CD." In these cases, the team assumed that aCustomer would receive response messages about the requested CD and inserted CD, respectively.

The fifth step was to add the execution occurrence to each object's and actor's lifeline. This was done by drawing a narrow rectangle box over top of the lifelines to represent when the objects (actors) are sending and receiving messages (e.g., in Figure 6-B aCustomer is active during the entire process, while aSR is only active at the beginning of the process (the top of the diagram)).

Finally, the CD Selections team validated the diagram by ensuring that the diagram accurately and completely represented the scenario of the Place Order use case being modeled. Figure 6-B portrays their completed sequence diagram.

## Communication Diagrams

Brian, one of the analysts, pointed out to the team that sequence diagrams and communication diagrams essentially modeled the same things. As such, he felt that it was not worth the time for the team to do both. And since they had already completed the sequence diagrams, he really did not want to do the communication diagrams also. However, even though the diagrams are very similar in what they portray, Alec decided that it would be worth the team's time to build both. He felt that it could be possible that the different formats of the diagrams might uncover additional requirements. As such, the team also created communication diagrams.

Alec chose to create the communication diagrams by following the steps to create communication diagrams described in the textbook. Like creating sequence diagrams, the first step in the process was to determine the context of the communication diagram. Alec chose to start with the same scenario of the Place Order use case that he and the team had used previously when they created the sequence diagrams (see Figure 6-B).

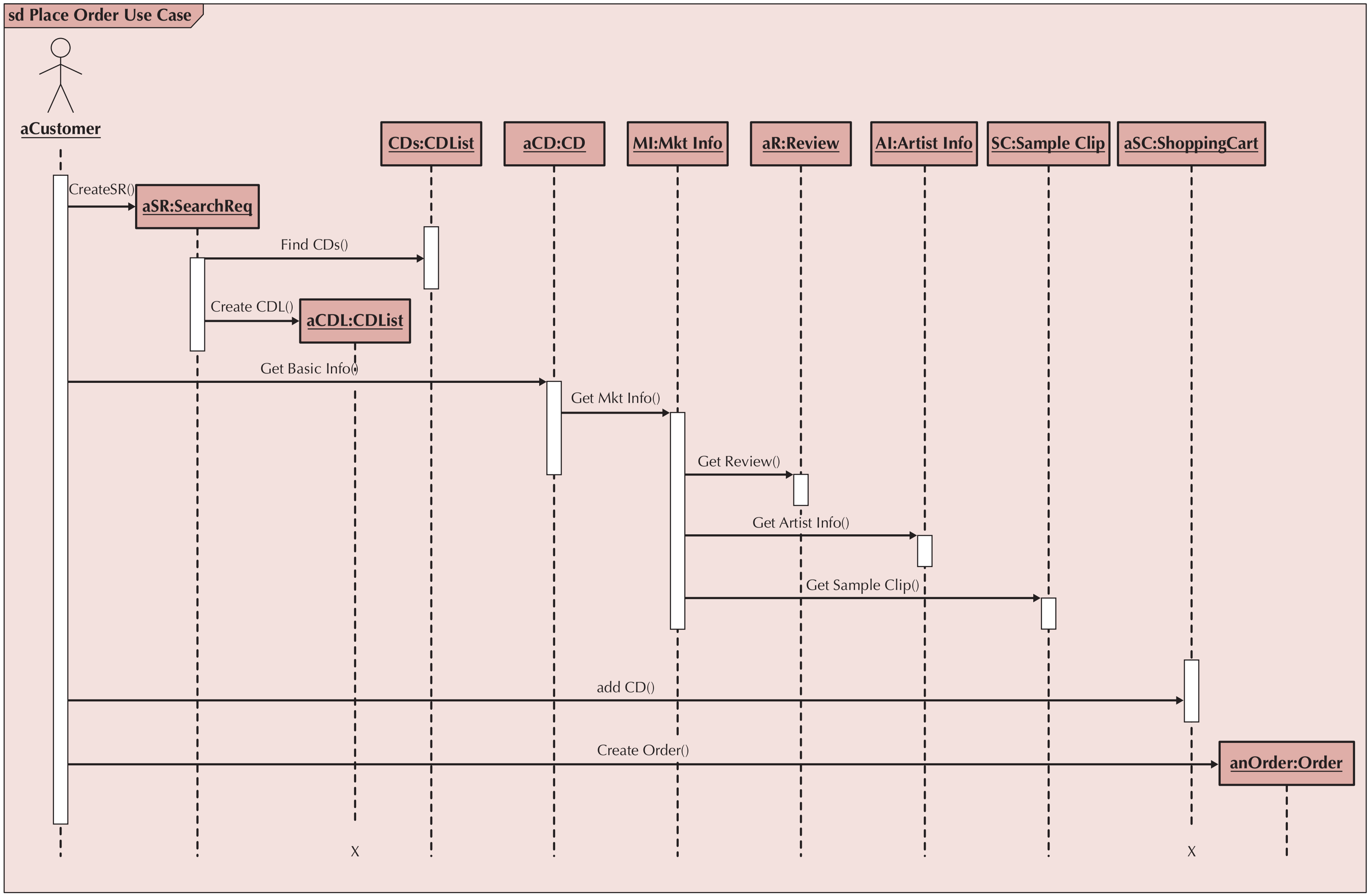

FIGURE 6-B Sequence Diagram for the Places Order Use Case

By executing the second step, the CD Selections team again identified the objects and the associations that link the objects together. Since they are using the same scenario as they did in the previously described sequence diagram, instances of the Search Request, CD List, CD, Marketing Material, Customer, Review, Artist Information, Sample Clip, and Shopping Cart classes should be the ones included. Also, the since the Customer actor interacts with the scenario, it should also be included. Furthermore, the team identified the associations between the objects (e.g., the instances of CD are associated with instances of Mkt Info, which, in turn, are associated with instances of Review, Artist Info, and Sample Clip).

During the third step, the team placed the objects on the diagram based on the associations that they have with the other objects in the collaboration. This was done to increase the readability, and hence, the understandability of the diagram (see Figure 6-C).

During the fourth step, the CD Selections team added the messages to the associations between the objects. For example, in Figure 6-C, the Create SR() message is the first message sent and the FindCDs() message is the second message sent. The aCustomer actor sends the Create SR() message to the aSR object, and the aSR object sends the FindCDs() message to the CDs object.

Finally, the CD Selections team executed the fifth and final step: validating the diagram. They accomplished this by ensuring that the scenario of the Place Order use case was accurately and completely represented by the diagram. See Figure 6-C for the completed communication diagram for this particular scenario of the Place Order use case. Furthermore, they compared the previously created sequence diagram (see Figure 6-B) with the communication diagram (see Figure 6-C) to ensure that that both diagrams were equivalent. The only difference between the two diagrams was the ease to portray the time ordering of the messages in the sequence diagram to represent how the objects interacted with each other in the communication diagram.

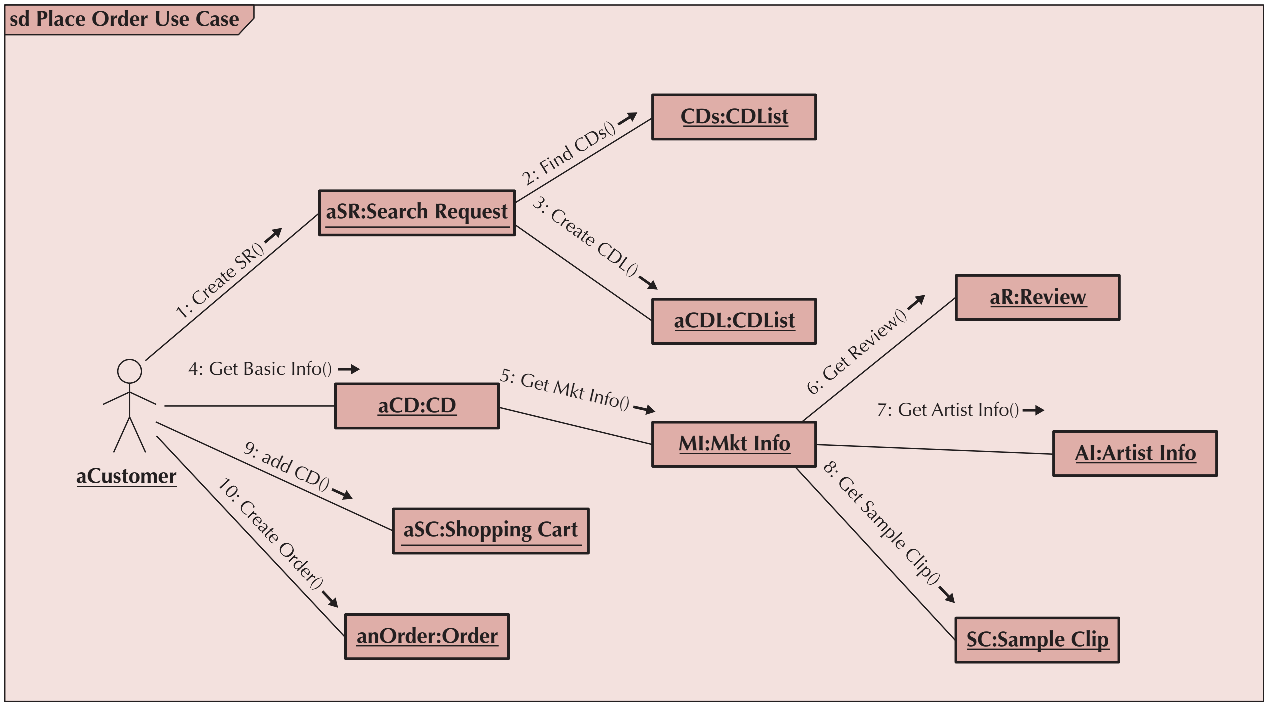

FIGURE 6-C Communication Diagram for the Places Order Use Case

## Behavioral State Machines

As in the previous example diagrams, we focus our attention only on the Place Order use case. Alec decided to follow the guidelines for creating a behavioral state machine (see Figure 6-10) and to follow the five steps for creating behavioral state machines described in the textbook. Like the earlier diagrams, the first step was to determine the context for the behavioral state machine to be drawn. Upon reviewing the objects involved in the scenario described by the sequence diagram (see Figure 6-A) and the communication diagram (see Figure 6-B) Alec decided that the team should focus on the Order class.

The second step was to identify the various states that an order will go through during its lifetime. To enable the discovery of the initial, final, and stable states of an order, Alec and the development team interviewed a customer representative that dealt with processing customer orders on a regular basis. Based on this interview, the team uncovered the life of an order (see Figure 6-D) from start to finish, from an order's perspective.

The third step is to determine the sequence of the states that an order object will pass through during its lifetime. Based on the order's lifecycle portrayed in Figure 6-D, the team identified and laid out the states of the order on the behavioral state machine.

Next, the team identified the events, actions, and guard conditions associated with the transitions between the states of the order. For example, the event "Order is created" moves the order from the "initial" state to the "In process" state (see Figure 6-E). During the "Processing" state, a credit authorization is requested. The guard condition "Authorization = Approved" prevents the order to move from the "Processing" state to the "Placed" state unless the credit authorization has been approved. Also, the guard condition "Authorization = Denied" prevents the order to move from the "Processing" state to the "Denied" state unless the credit authorization has been denied. As such, between the two guard conditions, the order is stuck in the processing state until the credit authorization has been either approved or denied.

The team finally validated the Order's behavioral state machine by ensuring that each state was reachable and that it was possible to leave all states except for the final states. Furthermore, the team made sure that all states and transitions for the order had been modeled. At this point in time, one of the analysts on the team, Brian, suggested that possibly there were multiple types of orders being described in the behavioral state machine. Specifically, he thought that there were denied and accepted orders. Based on this discovery, he suggested that two new classes, for each subtype of order, be created. However, upon further review by the entire team, it was decided that adding these classes to the class diagram and modifying all of the other diagrams to reflect this change would not add anything to the understanding of the problem. Therefore, it was decided not to add the classes. However, in many cases, modeling the states that an object will go through during its lifetime may in fact uncover additional useful subclasses. Figure 6-E illustrates the behavioral state machine that the CD Selections team created for an order object[5].

1. The customer creates an order on the Web.
2. The customer submits the order once he or she is finished.
3. The credit authorization needs to be approved for the order to be accepted.
4. If denied, the order is returned to the customer for changes or deletion.
5. If accepted, the order is placed.
6. The order is shipped to the customer.
7. The customer receives the order.
8. The order is closed.

FIGURE 6-D
The Life of an Order

[5]If the development team had more carefully read our textbook, they would have seen that they could have reused the Order behavioral state machine in Figure 6-17.

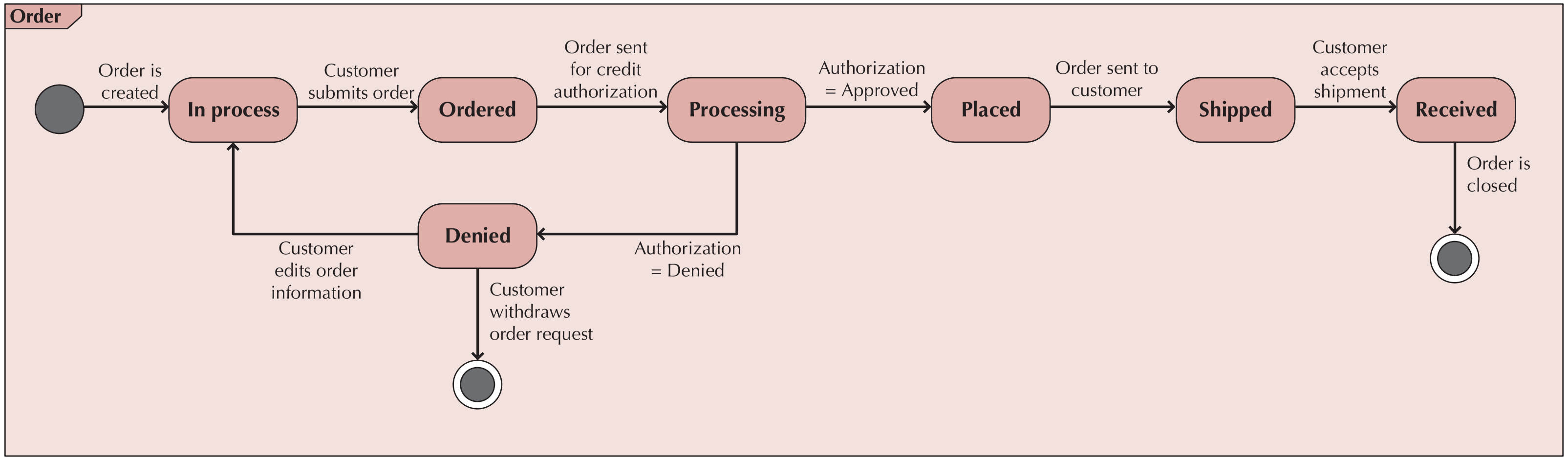

FIGURE 6-E Behavioral State Machine for the Order class

## CRUDE Matrix

As an attempt to tie the functional, structural, and behavioral models together, Alec decided to create a CRUDE matrix. To accomplish this, Alec assigned Anne the task of creating the matrix. As in the previous examples, we have limited this example to the Place Order use case.

To begin with, Anne created a class-by-class matrix. She then placed a (C)reate, (R)ead, (U)pdate, (D)elete, or a (E)xecute in each cell of the matrix that represented an interaction between instances of the classes. For example in Figures 6-B and 6-C, an instance of SearchReq created an instance of CDList. Also, in Figures 6-B and 6-C, an instance of CD references an instance of MktInfo. In this case, an "R" was placed in the CD:MktInfo cell. Figure 6-F shows the CRUDE matrix that Anne created based on the Place Order use case.

## Verifying and Validating the Behavioral Model

Once the team had completed all of the sequence diagrams, communication diagrams, behavioral state machines, and the CRUDE matrix, Alec had the team verify and validate the behavioral model. To accomplish this, Alec had the team use Figure 6-24 to identify where all of the common aspects of the different representations could be easily identified. For example, Alec pointed out that messages were not only contained in communication and sequence diagrams, but that they were also associated with the transitions in a behavioral state machines and the cell entries of the CRUDE matrix. Even though the team felt confident that the different representations were all consistent with each other, based on his past experience with teams that had short cut the verification and validation process, Alec insisted that all of the representations had to be verified and validated. Furthermore, Alec reminded them that they still needed to go back and modify the class diagram (see Figure 5-D) to include the Shopping Cart class. Consequently, the team still had quite a bit of work to do.

| | Customer | SearchReq | CDList | CD | Mkt Info | Review | Artist Info | Sample Clip | Shopping Cart | Order |
|---|---|---|---|---|---|---|---|---|---|---|
| Customer | | RU | | | | | | | U | C |
| SearchReq | | | CR | | | | | | | |
| CDList | | | | | | | | | | |
| CD | | | | | R | | | | | |
| Mkt Info | | | | | | U | U | U | | |
| Review | | | | | | | | | | |
| Artist Info | | | | | | | | | | |
| Sample Clip | | | | | | | | | | |
| Shopping Cart | | | | | | | | | | |
| Order | | | | | | | | | | |

FIGURE 6-F CRUDE Matrix for the Places Order Use Case