Object-Oriented Analysis Techniques

Use-Cases and Actor-Interaction Diagrams
Figure 5.17 Interaction diagram for the use case Serving Dinner. To make the diagram easier to understand, we have left out some of the objects.
THE PROCESS OF OBJECT MODELING

In performing object-oriented analysis (OOA), like any other systems analysis method, the purpose is to gain a better understanding of the system and its requirements. In other words, OOA requires that we identify the objects, their data attributes, associated behavior, and relationships that support the required business system requirements. We perform object modeling to document the identified objects, the data and behavior they encapsulate, plus their relationships with other objects.

There are two general activities when performing object-oriented analysis and they are as follows:

1. Finding and identifying the business objects.
2. Organizing the objects and identifying their relationships.

Finding and Identifying the Business Objects

In trying to identify objects, many methodology experts recommend searching the requirements specifications or other documentation and underlining the nouns that may represent potential objects. This could be a monumental task! There are just too many nouns.

One of the more popular and successful approaches for finding and identifying objects is a technique called use case modeling, developed by Dr. Ivar Jacobson.

Use case modeling is the process of identifying and modeling business events, who initiated them, and how the system responds to them.

Use case modeling provides a solution to this problem by breaking down the entire scope of system functionality into many smaller statements of system functionality called use cases. This smaller format simplifies and makes more efficient the technique of underlining the nouns. One advantage of use case modeling is that it identifies and describes the system functions from the perspective of external users. This is done by identifying and documenting events called use cases, which are initiated by users or systems called actors.

A use case is a behaviorally related sequence of steps (a scenario), both automated and manual, for the purpose of completing a single business task.

An actor represents anything that needs to interact with the system to exchange information. An actor is a user or a role that could be an external system or person.

An actor initiates system activity, a use case, to complete some business task. An actor represents a role fulfilled by a user interacting with the system and is not meant to portray a single individual or job title. Let’s use the example of a college student enrolling for the fall semester. The actor would be the student and the business event, or use case, would be enrolling in course. What about events that are triggered by time called temporal events? Who would be the actor? In the case of temporal events, the actor is the system itself. For example, on a nightly basis, a report is automatically generated listing which courses have been closed to enrollment (no open seats available) and which courses are still open. Notice that the report automatically gets generated every night; no one has to request that it be generated. This is a temporal event. The actor for this temporal event is the system itself.

Use cases are used during the entire system development process. During analysis, the use cases are used to model functionality of the proposed system and are the starting point for identifying the objects of the system. During the whole development process, use cases are continually refined in parallel with the process designing the objects. Because use cases contain an enormous amount of system functionality detail, they will be a constant resource for validating and testing development of the system design.

Use cases provide the following benefits:
Member Services Context Model

- As a basis to help identify objects and their high-level relationships and responsibilities.
- A view of system behavior from an external person's viewpoint.
- An effective tool for validating requirements.
- An effective communication tool.
- As a basis for a test plan.
- As a basis for a user's manual.

There are many approaches to begin use case modeling to help find and identify potential objects. They include prototyping, user and business analyst interviews, plus countless other fact-finding techniques. Usually it involves a combination of several fact-finding techniques. Let's now examine the steps involved in use case modeling to identify and find business objects for object modeling during systems analysis.

**Step 1: Identifying Actors and Use Cases** A good place to find potential actors and use cases is by analyzing the context model diagram of the system. Recall that a system context model illustrates the external parties that interact with the system to provide inputs and receive outputs. In doing so, it identifies the system's scope and boundaries. Figure 8.6 is a context model for the SoundStage Member Services System. Find the external parties that provide inputs to the system. Does the external party initiate the input or is the input a response to a request from the system? If the external party initiates the input, it is considered an actor. Some of
the inputs are self-explanatory, but others may be misleading. It is always wise to
confirm your findings with the system’s business analyst. In Figure 8.6, Club Mem-
ber is an actor that initiates and provides an input called Promotion Order. The
sequence of events in which the Member Services System would respond to the
card members Submitting a Promotion Order would be considered the use case.

Figure 8.7 presents our findings of analyzing the context diagram. It lists the
actors and the use cases they initiate.

Step 2: Constructing a Use Case Model  Once all the use cases and actors have been
identified, we have defined the total functionality of the entire system. A use case
model diagram can be used to graphically depict the system scope and bound-
aries in terms of use cases and actors. The use case model diagram for the Mem-
ber Services System is shown in Figure 8.8. It was created using Popkin Software’s
System Architect and represents the relationships between the actors and use cases
defined for each business subsystem. The subsystems represent logical functional
areas of business processes. The partitioning of system behavior into subsystems
is very important in understanding the system architecture and is key to defining
your development strategy—which use cases will be developed first and by whom.

Step 3: Documenting the Use Case Course of Events  For each use case identified, we
must now document the use case’s normal course of events. A use case’s normal
course of events is a step-by-step description starting with the actor initiating the
use case and ending with the business event. At this point we include only the
major steps that happen the majority of the time (its normal course). Exception
conditions or conditional branching logic will be documented later.

Figure 8.9 is a use case description for the Member Services System’s SUBMIT
PROMOTION ORDER use case. Notice it includes the following items:

A. The name of the actor who initiated the use case.
B. A high-level description of the use case.
C. A normal event course describing the use case’s major steps, from beginning
to end of this interaction with the actor.
D. Precondition describing the state the system is in before the use case is
executed.
E. Post-condition describing the state the system is in after the use case is
executed.
F. An assumptions section, which includes any nonbehavioral issues, such as
performance or security, that are associated with the use case but are difficult to model within the use case’s course of events.

**Step 4: Identifying Use Case Dependencies** Some use cases may be dependent on other use cases, with one use case leaving the system in a state that is a precondition for another use case. For example, a precondition of sending a club promotion is that the promotion must first be created. We use a diagram called the **use case dependency diagram** to model such dependencies. The use case dependency diagram provides the following benefits:

- A graphical depiction of the system’s events and their states enhances the understanding of system functionality.
- It helps to identify missing use cases. A use case with a precondition that is not satisfied by the execution of any other use case may indicate a missing use case.
- It facilitates project management by depicting which use cases are more critical (have the most dependencies) and thus need to have a higher priority.

Figure 8.10 represents the use case dependency diagram for the Member Services System. The use cases that are dependent on each other are connected with a dashed line labeled *depends on*. In Figure 8.10, the use case **SEND CLUB PROMOTION** has a dependency (precondition) on either the use case **CREATE NEW SEASONAL PROMOTION** or **CREATE NEW MONTHLY PROMOTION**.
### USE CASE

<table>
<thead>
<tr>
<th>USE CASE NAME:</th>
<th>Submit Promotion Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTOR:</td>
<td>Club Member</td>
</tr>
<tr>
<td>DESCRIPTION:</td>
<td>Describes the process when a club member submits a club promotion order to either indicate the products they are interested in ordering or declining to order during this promotion.</td>
</tr>
</tbody>
</table>
| NORMAL COURSE:| 1. This use case is initiated when the club member submits the promotion order to be processed.  
2. The club member's personal information such as address is validated against what is currently recorded in member services.  
3. The promotion order is verified to see if product is being ordered.  
4. The club member's credit status is checked with Accounts Receivable to make sure no payments are outstanding.  
5. For each product being ordered, validate the product number.  
6. For each product being ordered, check the availability in inventory and record the ordered product information which includes "quantity being ordered" and give each ordered product a status of "open."  
7. Create a Picking Ticket for the promotion order containing all ordered products which have a status "open."  
8. Route the Picking Ticket to the Warehouse. |
| PRECONDITION: | Use case *Send Club Promotion* has been processed. |
| POST-CONDITION:| Promotion order has been recorded and the Picking Ticket has been routed to the Warehouse. |

**FIGURE 8.9** Sample Use Case Description

---

**Step 5: Documenting the Use Case Alternate Course of Events**

During the previous steps of use case modeling, we focused on the normal courses of the use cases. This allowed us to concentrate on the system concept without getting bogged down in too many details. At this point the alternate courses and exception conditions in the use cases need to be defined. A use case has one normal event course that was previously defined, and possibly many alternate courses. These alternate courses are deviations, or branches, from the normal event course. Alternate courses are documented in a separate use case course. In Figure 8.11, the use case *SUBMIT PROMOTION ORDER* has several alternate courses. The first alternate course is for step 2 of the normal course. The club member has indicated an address or telephone change on the promotion order. Normally this does not occur, that's why it is documented as an alternate course. The analyst must refine each use case to include such alternate courses.

After you have defined the primary use cases with their normal and alternate courses, it is now time to start to identify the objects involved in the use cases. These objects represent things or entities in the business domain—things we are interested in and would like to capture information about. At this point we will concentrate on describing these objects with a sentence or two. Later we will
## USE CASE

**USE CASE NAME:** Submit Promotion Order

**ACTOR:** Club Member

**DESCRIPTION:** Describes the process when a club member submits a club promotion order to either indicate the products they are interested in ordering or declining to order during this promotion.

**NORMAL COURSE:**

1. This use case is initiated when the club member submits the promotion order to be processed.

2. The club member's personal information such as address is validated against what is currently recorded in member services.

3. The promotion order is verified to see if product is being ordered.

4. The club member's credit status is checked with Accounts Receivable to make sure no payments are outstanding.

5. For each product being ordered, validate the product number.

6. For each product being ordered, check the availability in inventory and record the ordered product information which includes "quantity being ordered" and give each ordered product a status of "open."

7. Create a Picking Ticket for the promotion order containing all ordered products which have a status "open."

8. Route the Picking Ticket to the Warehouse.

**ALTERNATE COURSE:**

2. If the club member has indicated an address or telephone number change on the promotion order, update the club member's record with the new information.

3. If the club member is not ordering product at this time, modify the promotion order's status to be "closed" and modify the selection of the month ordered product's record to have a status of "rejected," then cancel the transaction.

4. If Accounts Receivable returns a credit status that the customer is in arrears, invoke abstract use case Send Order Rejection Notice. Modify the promotion order's status to be "on hold pending payment."

5a. If the product number is not valid, create an Order Error Report containing the club member's information, the promotion order information, and the product number in error. Each completed report will be routed to a Member Services clerk for resolution.

5b. If the club member is not ordering the selection of the month, modify the ordered product's record to have a status of "rejected."

6. If the product being ordered is not available, record the ordered product information which includes "quantity being ordered" and give a status of "backordered."

7. If there are no ordered product records with a status of "open," cancel the transaction.

**PRECONDITION:** Use case Send Club Promotion has been processed.

**POST-CONDITION:** Promotion order has been recorded and the Picking Ticket has been routed to the Warehouse.

**ASSUMPTIONS:**

---

*FIGURE 8.11 Sample Use Case Description with Alternate Courses*
### USE CASE

**USE CASE NAME:** Submit Promotion Order  

**ACTOR:** Club Member  

**DESCRIPTION:** Describes the process when a club member submits a promotion order to either indicate the products they are interested in ordering or declining to order during this promotion.  

**NORMAL COURSE:**  

1. This use case is initiated when the club member submits the promotion order to be processed.  
2. The club member’s personal information such as address is validated against what is currently recorded in member services.  
3. The promotion order is verified to see if product is being ordered.  
4. The club member’s credit status is checked with Accounts Receivable to make sure no payments are outstanding.  
5. For each product being ordered, validate the product number.  
6. For each product being ordered, check the availability in inventory and record the ordered product information which includes “quantity being ordered,” and give each ordered product a status of “open.”  
7. Create a Picking Ticket for the promotion order containing all ordered products which have a status “open.”  
8. Route the Picking Ticket to the Warehouse.  

**ALTERNATE COURSE:**  

2. If the club member has indicated an address or telephone number change on the promotion order, update the club member’s record with the new information.  
3. If the club member is not ordering product at this time, modify the promotion order’s status to be “closed” and modify the selection of the month ordered product’s record to have a status of “rejected,” then cancel the transaction.  
4. If Accounts Receivable returns a credit status that the customer is in arrears, invoke abstract use case Send Order Rejection Notice. Modify the promotion order’s status to be “on hold pending payment.”  
5a. If the product number is not valid, create an Order Error Report containing the club member’s information, the promotion order information, and the product number in error. Each completed report will be routed to a Member Services clerk for resolution.  
5b. If the club member is not ordering the selection of the month, modify the ordered product’s record to have a status of “rejected.”  
6. If the product being ordered is not available, record the ordered product information which includes “quantity being ordered” and give a status of “backordered.”  
7. If there are no ordered product records with a status of “open,” cancel the transaction.  

**PRECONDITION:** Use case Send Club Promotion has been processed.  

**POST-CONDITION:** Promotion order has been recorded and the Picking Ticket has been routed to the Warehouse.  

**ASSUMPTIONS:**

---

**FIGURE 8.12** Sample Use Case Description with Nouns Highlighted
Use Case Diagram

The objective of use case modeling is to identify and define the business functions that require system support. The entire UP development process is based upon finding the use cases. This approach is called a use-case driven approach, and you will see that it drives the entire process (Jacobsen, 1992). We start the development of a use case diagram by identifying those people (or other systems) that will use the system. Those users are called actors. To ensure that we have the right level of abstraction, we will define actors only as those people (or things) that have direct contact with the automated system boundary. We emphasize that point by saying actors must have "hands." Next identify those core business functions done by each actor. A use case is a response that is done within the system in response to a request or input by a system user.

Figure 3 shows the actors and the use cases that we will include in the first iteration. We recognize that the video store system will need more use cases, but for the first iteration this will suffice. As you can see from the figure, we have decided that we need to have use cases to check out and return videos. We also need use cases to add movies to inventory. Finally we also need to add customers to our customer file. In later iterations we may expand the current use cases to be broader, such as removing movies and changing customers, but we keep it simple for the first iteration.

We emphasize that the actors in the diagram are those that are actually working with the system. Hence the checkout clerk is using the system to check out the movies for a customer. Since the end customer does not actually have contact with the automated system, it is not identified as an actor.

Use Case Detail Model

We identify this as separate from the use case diagram to emphasize that the use case model is only a high-level scoping model. It does not provide enough details to accurately describe the steps in the business processes and the required system responses. The use case detail model looks inside the oval of a use case to describe what is happening within the confines of a use case. One way to describe the internal steps is with a simple narrative description. Another way is to use another UML diagram called an activity diagram.

An activity diagram is a type of workflow diagram that is used to describe the sequence of steps that make up the use case. An activity diagram is made up of ovals, representing activities, and connecting lines, representing the flow from activity to activity. Vertical boxes, called swimlanes, are used to identify which actor does which activity.

Figure 4 is an example of an activity diagram for the Add new movies use case. Remember during analysis we are focusing on understanding the requirements of the new system. Essentially we are defining the steps in the business processes that interact with the system. Therefore this activity diagram only has two swimlanes, one for the
actor and one for the system. At this point we do not try to describe what is going on inside the system, just that it needs to do something to respond to the actor.

Figure 4. Activity diagram for Add New Movie use case.

One benefit of this approach is that the student is learning to focus on the business needs and the user activities. It also teaches about activity diagrams in a very simple context. At this point we expect the students to understand a use case diagram and a simple activity diagram. Students should also see the very close relationship between the two.
and that knowledge of its internals should not be needed by users of the object class. It should be clear from the operation's description whether it supplies the required functionality.

In this case a different approach is possible. The completed object classes can be regarded as black boxes. There is no need to explore the operations of a completed object class to show its internal workings. In these cases object interaction diagrams may not be needed, or you may choose only to expand, on the object interaction diagram, operations in the part of the system currently under development. Other uses for object interaction diagrams are for verifying that a transaction sequence can be supported by the existing object classes (in which case you choose to use a white box approach even for existing object classes) and for presentation/documentation purposes.

Activity flow diagrams can be useful during business analysis and logical design in combination with object interaction diagrams to clarify the logic flow of business processes and transaction sequences. They can also be used to clarify the logic of transaction sequences during system requirements analysis, when the standard and alternative paths are defined, although this is not their primary purpose.

Object Interaction Diagrams for the Seminar Registration System

We show two examples of fence-style object interaction diagrams. The first example shows a diagram produced during business analysis, modeling a black box view of the business process Seminar Registration to Invoicing. The second diagram shows an object interaction diagram for the transaction sequence Book Seminar, developed during logical design.

```
Figure 6-1: Seminar Registration to Invoicing - object interaction diagram
This diagram can be read a little like a spreadsheet. You start at the top left and read from left to right and from top to bottom.
```

1 Actors can also represent the business process tested and approved, and the actors involved in the transactions.
Across the top of the diagram, we show the actors who interact to produce the results required by the business process. In the diagram above, most of the actors are human. We also show our planned computer system as an actor - a machine interface labeled Administrative System. We are treating our planned computer system here as a black box. It is too early for us to consider how we might support the required functionality using object classes. Our objective during business analysis is to understand the process, not to specify computer support for it.

Associated with each actor, underneath the actor symbol, there is a vertical bar which represents the actor shown above the bar.

The arrows on the diagram represent requests which are sent by one actor (represented by one vertical bar) to another actor (represented by the vertical bar at the end of the arrow). In Figure 6-1, the first request is sent by the Employee to his or her Manager to request authorization to go to a seminar. We label the arrow with a name that explains the nature of the request from the sender’s point of view, e.g., Get manager’s authorization.

Notice that we show no return arrow for this request. Normally the actor who receives the request also responds to it by replying to the sender of the request. A less cluttered diagram results if you use a single arrow to show such a single request/response pair. One actor requests a service from another actor and we normally assume that that service is supplied and therefore don’t show it explicitly on the diagram.

The second request on the diagram is the request or message, Submit seminar booking request which the Employee sends to the Seminar Administrator. In this case, the actor who receives the request needs to send further requests to other actors before it can respond. In cases like this one, it can be clearer to show the request and the response to the request separately. We have chosen to do that with the response to the request labeled Notify employee.

The numbers shown on the left-hand side of the diagram represent links with an activity flow diagram and will be explained in the section on activity flow diagrams below.

---

1Actors can also be regarded as a type of object class. When we are modeling the business, we are modeling a system in the real world rather than a computer system. We could recognize the actors as objects and represent them using the object class symbol. However, it is friendlier and more intuitive to represent people as actors - especially on a diagram which may be shown to the people it represents.
An object interaction diagram for the transaction sequence Book Seminar

Our second object interaction diagram shows the planned computer system support for part of the business process described previously. The business process is supported by these transaction sequences:

- Book seminar
- Change seminar registration
- Pay invoice
- Cancel seminar registration.

Figure 6-2 is an object interaction diagram for the transaction sequence **Book seminar**. A description of this transaction sequence, with its standard and alternative paths, can be found in Chapter 5.
Figure 6.2: Book Seminar – object interaction diagram