Chapter 3 – Use Cases

Table of Contents

- Chapter Overview
- Learning Objectives
- Notes on Opening Case

Chapter Overview

This chapter extends the concepts learned in Chapter 2 about building analysis models that capture the processing requirements of the new system. For any system there are two types of requirements that must be defined and modeled: Processing requirements and data requirements. This chapter addresses the processing requirements showing you how to identify and document use cases. Use case modeling is a powerful technique to assist system analysts to identify, understand, communicate, and document the processing requirements of the new system.

The first section of the chapter teaches you how to find and identify use cases using the user goal technique. In this technique a systems analyst identifies the users of the system, as a role or type of user, and then identifies each goal or “actions to perform.” These goals then are used to define use cases.

The second method of identifying use cases is the “event-decomposition” technique. This technique first identifies the business events that occur. By understanding the business events, the actions leading up to the event, and the resulting processing required to support each event, a list of use cases can be developed. This is a powerful technique that takes a broader business point of view and can be used to identify many different types of events, which then produce a comprehensive list of use cases.

Another technique to help refine and verify events is the CRUD technique. CRUD, which stands for Create, Report, Update, Delete, is best used as a validation technique rather than a technique to find use cases. When you use CRUD to create a list use cases, you might produce very narrow use cases that frequently do not match the actual business processes. However, as a technique to validate an existing set of use cases, for example to ensure that the list is complete, it is quite effective. CRUD simply takes each problem domain class, e.g. database table, and ensure that there are sufficient use cases to create the date, update the data, delete the data, and report or query the data as appropriate.

The chapter concludes with instructions on how to build use case diagrams. Use case diagrams are a straightforward method to document and visualize the use cases that have been identified. Use case diagrams are effective in showing the actors, use cases, and relationships between use cases.

Note that this chapter does not get into the detailed descriptions or documentation of internal steps within a use case. That material is covered in Chapter 5.
Learning Objectives

After reading this chapter, the you should be able to:

- Explain why identifying use cases is the key to defining functional requirements
- Describe the two techniques for identifying use cases
- Apply the user goal technique to identify use cases
- Apply the event decomposition technique to identify use cases
- Apply the CRUD technique to validate and refine the list of use cases
- Describe the notation and purpose for the use case diagram
- Draw use case diagrams by actor and by subsystem

Notes on Opening Case

Opening Case

Waiters on Call Meal-Delivery System: This case shows how the analyst uses events—and the use cases they trigger—to summarize what the users require in their order and delivery system. As the users discuss their business, the analyst notes what events occur that require the system to respond in some way. A suggested in-class exercise is to complete a list of use cases.

Key points to note include:

- Events (and use cases) can be discovered by talking with the users.
- Talking about events is natural for business-oriented users.
- The analyst can focus on requirements independent of current or proposed technology (a logical model), which is something that you may find difficult. Even if the users want to talk about technology, it is important to define the requirements in this way.

Use Cases and User Goals

Key Terms

- **use case** – an activity that the system performs, usually in response to a request by a user
- **user goal technique** – a technique to identify use cases by determining what specific goals or objectives must be completed by a user

One approach to identifying use cases, called the user goal technique, is to ask users to describe their goals for using the new or updated system. The analyst first identifies all the users and then conducts a structured interview with each user. A user goal can be thought of as a piece of work, or a task, that the user must complete. For example, for an RMO a shipping clerk might have a goal such as *ship items*, or *track shipments*. 
The user goal technique for identifying use cases includes these steps:

1. Identify all the potential users for the new system.
2. Classify the potential users in terms of their functional role (e.g., shipping, marketing, sales).
3. Further classify potential users by organizational level (e.g., operational, management, executive).
4. For each type of user, interview them to find a list of specific goals they will have when using the new system. Start with goals they currently have and then get them to imagine innovative functions they think would add value. Encourage them to state each goal in the imperative verb-noun form, such as Add customer, Update order, and Produce month end report. Create a list of preliminary use cases organized by type of user.
5. Look for duplicates with similar use case names and resolve inconsistencies.
6. Identify where different types of users need the same use cases.
7. Review the completed list with each type of user and then with interested stakeholders.

**Use Cases and Event Decomposition**

**Key Terms**

- **event decomposition technique** – a technique to identify use cases by determining the external business events to which the system must respond
- **elementary business processes (EBPs)** – the most fundamental tasks in a business process, which leaves the system and data in a quiescent state; usually performed by one person in response to a business event
- **event** – something that occurs at a specific time and place, can be precisely identified, and must be remembered by the system
- **external event** – an event that occurs outside the system, usually initiated by an external agent
- **actor** – an external agent; a person or group that interacts with the system by supplying or receiving data
- **temporal event** – an event that occurs as a result of reaching a point in time
- **state event** – an event that occurs when something happens inside the system that triggers some process
- **system controls** – checks or safety procedures to protect the integrity of the system and the data
- **perfect technology assumption** – the assumption that a system runs under perfect operating and technological conditions

The most comprehensive technique for identifying use cases is the event decomposition technique. The event decomposition technique begins by identifying all the business events that will cause the information system to respond, and each event leads to a use case. Starting with business events helps the analyst define each use case at the right level of detail. The appropriate level of detail for identifying use cases is one that focuses on elementary business processes (EBPs). An EBP is a task
that is performed by one person in one place in response to a business event, adds measurable business value, and leaves the system and its data in a stable and consistent state.

Event Decomposition Technique

The event decomposition technique focuses on identifying the events to which a system must respond and then determining how a system must respond (i.e., the system’s use cases). When defining the requirements for a system, it is useful to begin by asking, “What business events occur that will require the system to respond?”

Types of Events

External Events: An external event is an event that occurs outside the system—usually initiated by an external agent or actor. An external agent (or actor) is a person or organizational unit that supplies or receives data from the system. To identify the key external events, the analyst first tries to identify all the external agents that might want something from the system. A classic example of an external agent is a customer. When describing external events, it is important to name the event so the external agent is clearly defined. The description should also include the action that the external agent wants to pursue.

Temporal Events: A temporal event is an event that occurs as a result of reaching a point in time. For example, on month-end certain automatic processing may need to be initiated. The analyst begins identifying temporal events by asking about the specific deadlines that the system must accommodate. What outputs are produced at that deadline? What other processing might be required at that deadline?

State Events: A state event is an event that occurs when something happens inside the system that triggers the need for processing. For example a state event can occur when an inventory level reaches a pre-order point.

Identifying Events

Events versus Prior Conditions and Responses: It is sometimes difficult to distinguish between an event and part of a sequence of prior conditions that leads up to the event. The way to determine whether an occurrence is an event or part of the interaction following the event is by asking whether any long pauses or intervals occur (i.e., can the system transaction be completed without interruption?). Or is the system at rest again, waiting for the next transaction? After the customer wants to buy the shirt, the process continues until the transaction is complete. There are no significant stops after the transaction begins. After the transaction is complete, the system is at rest, waiting for the next transaction to begin. The EBP concept defined earlier describes this as leaving the system and its data in a consistent state.

The Sequence of Events: Tracing a Transaction's Life Cycle: It is often useful in identifying events to trace the sequence of events that might occur for a specific external agent or actor. This is similar to the user goal technique, in that it focuses on a single actor. But it focuses on a narrower set of tasks. For example, what are all the things that a customer might want to do when he or she buys a shirt? Such things as buy the shirt, return the shirt, exchange the shirt, and so forth.

Technology-Dependent Events and System Controls: Sometimes, the analyst is concerned about
events that are important to the system but do not directly concern users or transactions. Such events typically involve design choices or system controls. These types of events are not part of the problem domain, e.g. not part of the users requirements. Typically those events are not addressed during analysis but are addressed during the design activities. One technique used to help decide which events apply to controls is to assume that technology is perfect. The perfect technology assumption states that events should be included during analysis only if the system would be required to respond under perfect conditions.

Using the Event Decomposition Technique

To summarize, the event decomposition technique for identifying use cases includes these steps:

1. Consider the external events in the system environment that require a response from the system by using the checklist shown in Figure 3-3.
2. For each external event, identify and name the use case that the system requires.
3. Consider the temporal events that require a response from the system by using the checklist shown in Figure 3-4.
4. For each temporal event, identify and name the use case that the system requires and then establish the point of time that will trigger the use case.
5. Consider the state events that the system might respond to, particularly if it is a real-time system in which devices or internal state changes trigger use cases.
6. For each state event, identify and name the use case that the system requires and then define the state change.
7. When events and use cases are defined, check to see if they are required by using the perfect technology assumption. Do not include events that involve such system controls as login, logout, change password, and backup or restore the database, as these are put in as system controls.

Use Cases and CRUD

Key Terms

- CRUD technique – an acronym for Create, Read/Report, Update, and Delete; a technique to validate or refine use cases

Another important technique used to validate and refine use cases is the CRUD technique. “CRUD” is an acronym for Create, Read or Report, Update, and Delete, and it is often introduced with respect to database management. The CRUD technique is most useful when used as a cross-check along with the user goal technique.

The CRUD technique for validating and refining use cases includes these steps:

- Identify all the data entities or domain classes involved in the new system. Chapter 4 discusses these in more detail.
- For each type of data (data entity or domain class), verify that a use case has been identified that
creates a new instance, updates existing instances, reads or reports values of instances, and deletes (archives) an instance.

- If a needed use case has been overlooked, add a new use case and then identify the stakeholders.
- With integrated applications, make sure it is clear which application is responsible for adding and maintaining the data and which system merely uses the data.

Use Cases in the Ridgeline Mountain Outfitters Case

Key Terms
- **brief use case description** – an often one-sentence description that provides a quick overview of a use case.

The initial system vision (discussed in Chapter 2) identified four subsystems: the Sales subsystem, the Order Fulfillment subsystem, the Customer Account subsystem, and the Marketing subsystem. As work progressed, the analysts combined reports required by each subsystem into a fifth subsystem called the Reporting subsystem. In a system this size, the analyst should organize the use cases by subsystem to help track which subsystem is responsible for each use case.

These use cases are shown in Figures 3-10 and 3-11. It is important to recognize that the list of use cases will continue to evolve as the project progresses. This **brief use case description** is usually expanded to record more of the details when the developers are designing and implementing the use case.

Use Case Diagrams

Key Terms
- **use case diagram** – the UML model used to graphically show use cases and their relationships to actors.
- **automation boundary** – the boundary between the computerized portion of the application and the users who operate the application but are part of the total system.
- **«includes» relationship** – a relationship between use cases in which one use case is stereotypically included within the other use case.

The use case diagram is the UML model used to graphically show the use cases and their relationship to users. In UML, a person that uses the systems is called an actor. An actor is always outside the automation boundary of the system but may be part of the manual portion of the system. Sometimes, the actor for a use case is not a person; instead, it can be another system or device that receives services from the system.

A simple stick figure is used to represent an actor. The stick figure is given a name that characterizes the role the actor is playing. The use case itself is represented by an oval with the name of the use case inside. The connecting line between the actor and the use case indicates that the actor is involved with that use case. Finally, the automation boundary, which defines the border between the computerized...
portion of the application and the people operating the application, is shown as a rectangle containing the use case. Figures 3-12 through 3-16 give several examples of use case diagrams.

There are many ways to organize use case diagrams for communicating with users, stakeholders, and project team members. One way is to show all use cases that are invoked by a particular actor or functional area such as a department.

Frequently during the development of a use case diagram, it becomes apparent that one use case might use the services of another use case. Therefore, one use case uses, or “includes,” another use case. Sometimes, this relationship is referred to as the «includes» relationship or the «uses» relationship. Note that the word “includes” is enclosed within guillemets in the diagram.

Developing a Use Case Diagram

The steps to develop use case diagrams are:

1. Identify all the stakeholders and users who would benefit by having a use case diagram.
2. Determine what each stakeholder or user needs to review in a use case diagram. Typically, a use case diagram might be produced for each subsystem, for each type of user, for use cases with the includes relationship, and for use cases that are of interest to specific stakeholders.
3. For each potential communication need, select the use cases and actors to show and draw the use case diagram. There are many software packages that can be used to draw use case diagrams.
4. Carefully name each use case diagram and then note how and when the diagram should be used to review use cases with stakeholders and users.

Final Thoughts

Probably the biggest problem that you may have with the concepts in this chapter is finding and describing use cases. A big help is for you is to say “the actor uses the system to .... [use case description].” For example, “the customer uses the system to 'make a purchase’” or 'fill a shopping cart.' Using this technique most frequently describes use cases at the right level of detail.

Also remember that a use case should stand alone. In other words, the system should be in an inactive state when the use case starts and it should end in an inactive state. For example, one type of action could be an independent use case, or that same action might exist only as one step in a use case. Take, for our example, “make a payment.” If all purchases must be paid when the purchase is made, then making a payment is always part of the “purchase an item” use case. However, if the organization allows customer lines of credit, then making a payment may also be an independent use case, e.g. the system is quiescent before and after making a payment. In this case, “purchase an item” might «include» “make a payment” use case.

The other problem that you may have is use CRUD to define all of your use cases. Using CRUD as the sole source of use cases results in a lot of low level, non-business descriptions for use cases. CRUD is best used as a validating technique and not a use case definition technique.