Chapter 5 – Use Case Modeling

Table of Contents

Chapter Overview

The main objective of defining requirements in system development is understanding users' needs, how the business processes are carried out, and how the system will be used to support those business processes. Model building is one of the best ways for analysts to understand the business processes and to remember what they have learned from their fact finding activities.

Use case related models are the "fully developed use case description," UML activity diagrams, and UML system sequence diagrams (SSD). The "fully developed use case descriptions" are used to document the context, purpose, description, conditions, and workflow of each use case. Activity diagrams are a graphical depiction of the use case workflow and are useful in illustrating the alternative paths through a business process. SSDs are used to document the inputs and outputs that are passed between the user and the system during a use case. Not only do these models document the internal steps of a use case, but the very act of developing these models force the analyst to ask detailed questions and help improve the understanding of the requirements.

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 students use CRUD to create a list use cases, they usually 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.

©2016. Cengage Learning. All rights reserved.

Learning Objectives

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

- Write fully developed use case descriptions
- Develop activity diagrams to model flow of activities
- Develop system sequence diagrams
- Use the CRUD technique to validate use cases
- Explain how use case descriptions and UML diagrams work together to define functional requirements

Notes on Opening Case and EOC Cases

Opening Case

Electronics Unlimited: Integrating the Supply Chain: This case is a short description of the need for businesses now to integrate their own systems, particularly the input/output interfaces with systems from other companies. To have a truly integrated supply chain, information must be passed between companies. The object-oriented models, as defined by UML, provide a strong basis to document interface requirements. Object-oriented system development provides the techniques to actually pass data between companies. (The other important technique is XML, which is not discussed in this case.)

EOC Cases

TheEyesHaveIt.com Book Exchange: TheEyesHaveIt.com Book Exchange is a type of e-business exchange that does business entirely on the Internet. The company acts as a clearinghouse for buyers and sellers of used books. The case describes the data and processing requirements. Students are asked to develop domain model, list of use cases, detailed use case descriptions, and System Sequence Diagrams (SSD). This case reviews material from Chapters 3 and 4, as well as the new material in Chapter 5.

Community Board of Realtors (running case)**:** Community Board of Realtors is a professional organization that supports real estate offices and agents. The description in this chapter refers back to Chapters 3 and 4. Students are to create fully developed use case descriptions and SSDs for two use cases. Students are also asked to develop a CRUD analysis matrix.

Spring Breaks 'R' Us Travel Services (SBRU) (running case)**:** SBRU is an online travel services that books spring break trips to resorts for college students. The description in this chapter refers back to Chapters 3 and 4. Students are to create fully developed use case descriptions and SSDs for two use cases. Students also develop an activity diagram and a CRUD analysis.

©2016. Cengage Learning. All rights reserved.

On the Spot Courier Services (running case)**:** On the Spot is a small, but growing, courier service that needs to track customers, package pickups, package deliveries, and delivery routes. The description in this chapter extends the basic processes to include various types of real-time, online, and web-based capabilities. This chapter will require the students to reuse the skills they developed in earlier chapters. Students are then asked to create fully developed use case descriptions, activity diagrams, SSDs, and a CRUD analysis.

Sandia Medical Devices (running case)**:** Sandia Medical Devices is a company that specializes in medical monitoring through remote, mobile telecommunication devices. The description in this chapter has an existing use case diagram that the students must analyze and understand. Students are also asked to develop an SSD for one use case.

Instructor's Notes

Use Case Descriptions

Key Terms

- **use case description** – a textual model that lists and describes the processing details for a use case
- **scenarios or use case instances –** unique sets of internal activities within use cases
- **precondition –** a condition that must be true before a use case begins
- **postcondition –** what must be true upon the successful completion of a use case

Lecture Notes

Foundation

A **use case description** lists and describes the processing details for a use case. Implied in all use cases is a person who uses the system. In UML, that person is called an actor, as shown on use case diagrams. Another way to think of an actor is as a role.

Internally, a use case includes a whole sequence of steps to complete a business process. Frequently, several variations of the business steps exist within a single use case. These different flows of activities are called **scenarios** or sometimes **use case instances**. Thus, a scenario is a unique set of internal activities within a use case and represents a unique path through the use case.

A use case description provides the documentation for the processing steps that are internal to a use case. There may be several descriptions, each one describing a different scenario. Depending on an analyst's needs, use case descriptions tend to be written at two separate levels of detail: brief description and fully developed description. As shown later in the chapter, an activity diagram is also a useful technique to describe the internal processing of a use case.

©2016. Cengage Learning. All rights reserved.

Brief Use Case Descriptions

A brief description can be used for very simple use cases, especially when the system to be developed is a small, well-understood application. A simple use case would normally have a single scenario and very few—if any—exception conditions. Figure 5-1 illustrates several single sentence brief descriptions. A brief description does not document the internal processing since for simple use cases there are few steps.

Fully Developed Use Case Descriptions

The fully developed description is the most formal method for documenting a use case. The following items document each row of the example shown in Figure 5-2.

1. Use case name: a unique name for this use case
2. Scenario: the instance of the use case being documented
3. Triggering Event: the business event that initiates or triggers this use case
4. Brief Description: a one or two sentence description of the results of the use case
5. Actors: all those actors who use this use case
6. Related use case: any included use cases, or use cases that "include" this one
7. Stakeholders: other persons, other than the actors, who have an interest in the results or successful completion of this use case
8. **Precondition**: The state of the system before the use case begins, especially any required conditions of the database
9. **Postcondition**: the state of the system at the successful completion of the use case, especially any updates to the database
10. Flow of activities: a step by step sequence of the actions by the actor and the system internal to the use case
11. Exception conditions: any exception conditions that cause the system not to follow the expected flow of activities or that cause the use case to terminate abnormally

Usually the flow of activities is the most difficult part to develop, but also assists the analyst and the user to understand the requirements more deeply.

Quick Quiz

Q: When would you use a brief use case description as opposed to a fully developed one?

A: Either as preliminary documentation, or for a very simple use case where the processing steps are few and obvious.

Q: What is meant by a "precondition?"

A: A precondition defines the state of the system, including the database, before a use case begins. Often it identifies conditions that must be true in order for the use case for fire

©2016. Cengage Learning. All rights reserved.

successfully.

Q: What is meant by a “postcondition?”

A: A postcondition describes the state of the system after the use case has successfully completed. It includes any updates to the database that must be remembered, e.g. persistent classes that are updated.

Activity Diagrams for Use Cases

Key Terms

Activity diagrams were introduced in Chapter 2 as a technique for documenting user workflows. See Chapter 2 for terms.

Lecture Notes

An activity diagram is an easily understood diagram to document the workflows of the business processes. Activity diagrams are a standard UML diagram that are also an effective technique to document the flow of activities within a use case. Activity diagrams are helpful when the flow of activities for a use case is complex.

Remember that an activity diagram used rounded squares to identify individual activities within the workflow. It is also possible to include ovals, which represented included use cases, within a workflow. Figure 5-6 illustrates the *Fill shopping cart* use case, with three other use cases included as part of the total workflow.

Quick Quiz

Q: Why are activity diagrams useful for understanding a use case?

A: An activity diagram documents the step by step processing within a use case and shows the relationships (arrows) between the steps. It is also graphical and consequently often easier to understand.

The System Sequence Diagram – Identifying Inputs and Outputs

Key Terms

- **system sequence diagram (SSD)** – a diagram showing the sequence of messages between an external actor and the system during a use case or scenario
- **lifeline or object lifeline** – the vertical line under an object on a sequence diagram to show the passage of time for the object
- **loop frame** – notation on a sequence diagram showing repeating messages
- **true/false condition** – part of a message between objects that is evaluated prior to transmission to determine whether the message can be sent

©2016. Cengage Learning. All rights reserved.

- **opt frame** – notation on a sequence diagram showing optional messages
- **alt frame** – notation on a sequence diagram showing if-then-else logic

Lecture Notes

In the object-oriented approach, the flow of information is achieved through sending messages either to and from actors or back and forth between internal objects. A **system sequence diagram (SSD)** is used to describe this flow of information into and out of the automated system, e.g. between the actor and the system. Often a SSD is used in conjunction with an activity diagram to get a complete picture of the steps and information flows of a use case.

System Sequence Diagram (SSD) Notation

In a use case diagram, the actor "uses" the system, but the emphasis in an SSD is on how the actor "interacts" with the system by entering input data and receiving output data. Refer to figure 5-7 and 5-8 to note the basic notation used in an SSD. A stick figure represents an actor. The box labeled :System is an object that represents the entire automated system. (UMO object notation is similar to UML class notation, except the object name is often preceded by a colon and is underlined.)

Connected to the actor and the :System object are dashed lines, which represent **life lines** or **object lifeline**. Messages are passed between the actor and the :System and are assumed to be passed sequentially top to bottom along the life lines.

A message is documented with an arrow and a message descriptor. Since a SSD is a UML object-oriented model, the syntax for the messages is similar to interaction, e.g. programming methods, syntax that is found in various OO programming languages. Here is the notation for the message descriptor:

**[true/false condition] return-value := message-name (parameter-list)*

Any part of the message can be omitted. In brief, the notation components do the following:

- An asterisk (*) indicates repeating or looping of the message.
- Brackets [ ] indicate a true/false condition. This is a test for that message only. If it evaluates to true, the message is sent. If it evaluates to false, the message isn't sent.
- Message-name is the description of the requested service. It is omitted on dashed-line return messages, which only show the return data parameters.
- Parameter-list (with parentheses on initiating messages and without parentheses on return messages) shows the data that are passed with the message.
- Return-value on the same line as the message (requires :=) is used to describe data being returned from the destination object to the source object in response to the message. A return value may also be shown as a separate return message on a dashed line.

To show a more complex flow of activities, **loop frames**, **opt frames**, and **alt frames** are used. The frame is a rectangle which encloses the messages that are part of the loop or opt or alt. The rectangle is labeled as a loop (looping set of messages), opt (optional messages), or alt (if-then-else messages).

©2016. Cengage Learning. All rights reserved.

Developing a System Sequence Diagram (SSD)

To develop an SSD, it is useful to have a detailed description of the use case—either in the fully developed form or as an activity diagram. One advantage of using activity diagrams is that it is easy to identify when an input or output occurs. Inputs and outputs occur whenever an arrow in an activity diagram goes from an external actor to the computer system.

Recall for a workflow diagram there are two swimlanes: the actor and the computer system. The system boundary coincides with the vertical line between the actor's swimlane and the computer system's swimlane. The development of an SSD based on an activity diagram falls into four steps:

1. Identify the input messages, which are the transition arrows that cross the system boundary.
2. Describe the message from the external actor to the system by using the message notation described earlier. The message name should describe the service requested from the system, such as *create a customer*. Include the parameters that the system will need to carry out the requested service.
3. Identify and add any special conditions such as iteration and true/false conditions.
4. Identify and add the outputs or return values from each message.

This sequence of steps two through four are done for each message identified in step one. One of the advantages of creating a SSD is that it helps the analyst understand and verify not only the processing steps of the use case, but the data values that are required.

Upon completion of this activity, the analyst will have a full description of a use case, with its workflow in an activity diagram, and good documentation of the inputs and outputs required. Model building is a powerful tool to help analysts understand user requirements. It also provides a comprehensive document that can be used in systems design and programming.

Quick Quiz

Q: What is the relative benefit of an activity diagram and an SSD?

> A: An activity diagram helps the analyst understand the flow of the work, and the SSD describes the associated inputs and outputs that occur during the workflow.

Q: What are the component parts of a message notation?

> A: Message name, parameter list in parenthesis, return values, true/false condition, iteration asterisk.

Q: What is the difference between swimlanes and life lines? Both notation and use?

> A: Swimlanes are long rectangles and contain the activities which correspond to the actor or system. Sequence of activities is denoted by arrows. Life lines are vertical dashed lines which also correspond to the actor or system. Sequence of messages is read from top to bottom.

©2016. Cengage Learning. All rights reserved.

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

Lecture Notes

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.

Quick Quiz

Q: What is the best or most useful way to use the CRUD technique?

A: CRUD is best used as a check or validation of existing use cases. When CRUD is used to identify use cases, often the use cases are too low level and do not reflect the business processes nor the user goals.

Q: What does a CRUD matrix or table show?

A: A CRUD table shows the data entities or domain classes across the top, with a list of the use cases along the right side. Those use cases that either create, report, update, or delete the particular data entity are noted in the cells of the table.

Integrating Requirements Models

Key Terms

none

©2016. Cengage Learning. All rights reserved.

Lecture Notes

At this point, students have learned about and learned how to create several requirements models. One of the key concepts for students to understand is that all of these models link together to provide a complete picture of the user's requirements. And, it is extremely important that the models are consistent with each other. Information across all of these models must be consistent and provide a unified picture of the requirements.

Two diagrams are especially important because they provide an overview or comprehensive view of the entire system. One is the use case diagram, which in its complete form identifies all of the use cases to be implemented. The other is the class diagram or the domain model, which provides information about all of the data items required.

As shown in Figure 5-14, other diagrams feed off of these two diagrams. The Use case diagram feeds into the Use case descriptions, Activity diagrams, and SSDs, all of which document information about individual use cases. The State machine diagram and to some extent the SSD are fed by the class diagram. A state machine diagram covers only one class, where the SSD may refer to many different objects or attributes as parameters in the messages.

Quick Quiz

Q: Which are the two overview or high-level diagrams?

A: Use case diagram and the domain model class diagram.

Q: A single use case is described in depth by which models?

A: Fully developed use case description, activity diagram, SSD

Q: The life of the objects in a single object class is described by which model?

A: A state transition machine

Classroom Activities

Since this chapter is primarily focused on developing diagrams and models, the best way to teach this material is by doing class examples. It is usually ineffective just to explain and show the models. Being able to develop models is a skill that is only learned by doing. If time allows, the best approach is to develop a sample model together. The teacher can take a case or one of the problems and develop the solution while thinking out loud and commenting on each portion. A second example can then be done by first assigning it to the students, and giving them 5-10 minutes to work on the solution, and then again, completing the solution as an in-class example.

Usually both the use case diagram and an activity diagram are not too hard for the students since they have previous experience. The development of a SSD will require more detailed explanations and examples.

©2016. Cengage Learning. All rights reserved.

Troubleshooting Tips

Fully Developed Use Case Description

Students sometimes have difficulty understanding the difference between a use case and a scenario. A use case is the generalized version, and a scenario is a particular instance of it. The best way to explain this concept is with an example. A generalized use case of *Purchase merchandise* is different for shopping online than it is for shopping in a store. The online version has many more computer system related steps to find merchandise, show images of the merchandise, add it to the shopping cart, and then checking out. The store version does not involve the computer system until it is time to check out. So even though the general use case is the same, the internal processing flow is very different. Hence, each version must be documented separately.

Students also sometimes have a hard time understanding pre-conditions and post-conditions. Almost all pre-conditions and post-conditions have to do with the data in the database, its availability, and its condition. Again, the best way to teach it is through examples. In order for the *Purchase merchandise* use case to even begin, there must be merchandise available. Depending on the user’s definition, a customer data record may also need to be in existence. These are pre-conditions.

After the use case has finished executing, then there must also be a reduction in merchandise inventory, and there must be a purchase object in a particular status (such as ready for shipping). Depending on the requirements there must either be a payment object, or perhaps an increase in the customer account value. You may have to have the students do a couple of examples to understand pre-conditions and post-conditions.

The Flow of Activities is also sometimes confusing. The idea is simply that the user does something, the system responds. It goes back and forth. For many students, especially those that are visually oriented, it may be easier to develop an activity diagram rather than the Flow of Activities. Both essentially capture the same information.

Activity Diagrams

Students have already used activity diagrams in documenting user workflows. We find that activity diagrams are fairly easy concepts for students to grasp. The two swimlanes, between the user and the system, keep the activity diagram simple.

Probably the major area of difficulty is how detailed to make the activity ovals. One extreme has the ovals almost at the keystroke level, while the other extreme can extend to the entire use case. First, students need to remember the “Prefect technology assumption” and try to remove those activities. Then each activity should be a logical action, such as “search for a product” or at a little lower level, “enter product information for search.” The activities need to be at a low enough level to show the interactions between the user and the system.

One potential danger to avoid is to keep the examples and problems simple. In today's Web based systems with hotlinks and hyper-links in abundance, the users can take a multitude of alternative paths through a process. Be sure to keep your examples focused, such as only to find merchandise and buy it. Avoid all of the other things that users can do to bounce around the entire site.

©2016. Cengage Learning. All rights reserved.

System Sequence Diagram

Sequence diagrams are one of the more difficult UML models to learn. Sequence diagrams are a powerful design model that is taught in depth in Chapter 11. In order to facilitate learning and help students understand sequence diagrams, we introduce sequence diagrams with the more simplified system sequence diagram (SSD) version. However, an SSD is a powerful analytical model in its own right and provides a good insight into use case internals.

There are many parts to an SSD for a student to learn. Be sure to take the time so that they understand the concept of a lifeline (simply the life of an object, which is read top to bottom), and a message (from a programming viewpoint it is the invoking of a method, or a method call, with data parameters being passed). The objective of the SSD is to document the actual input data and the output data that flows between the user actor and the system. The easiest way to identify what messages are required is to use the activity diagram of the use case. Every time an arrow crosses the center boundary line, there is an input or output message.

Discussion Questions

Most of this chapter is teaching modeling skills. It is recommended that most of the class time be spent on in-class examples and skill development. However, the following discussion question is important.

1. Integration of requirements models

Students frequently underestimate the integrated nature of the UML models and do not grasp how the models all integrate together to give a total description of the business requirements. What does it mean to have integrated requirements models? Explain how each of the models relate to the other models – models we have learned are use case diagram, CRUD matrix, class diagram, activity diagram, fully developed use case diagram, system sequence diagram, state machine diagram. How can cross checking between models ensure that the requirements are accurate and correct? What are the dangers of not cross checking between the models?

©2016. Cengage Learning. All rights reserved.