CHAPTER 8

Activity Diagrams

MAJOR TOPICS

OBJECTIVES

At the completion of this chapter, you will be able to:

- Describe the benefits of activity diagrams.
- Interpret activity diagrams.
- Create activity diagrams to describe use-cases.

PRE-TEST QUESTIONS

The answers to these questions are in Appendix A at the end of this manual.

1. What is the primary purpose of Activity Diagrams?

....

....

INTRODUCTION

A use-case diagram describes the relationships between actions and discrete units of a system's functionality. A use-case description provides a brief overview of the purpose of each use-case and the steps required to complete that purpose. An activity diagram can be used to expand on a use-case description. Activity diagrams are similar to flow charts: they describe the order of activity and the branch logic of a process. However, they differ from traditional flow charts by allowing the representation of concurrent operations. Activities that take place simultaneously (such as threads) can be represented using activity diagrams. Activity diagrams can be used to supplement the use-case descriptions within a use-case model.

Creating activity diagrams

Activity diagrams flow from top to bottom. The initial state is represented by a closed circle. Activity proceeds through a series of activity states until it reaches its final state, which is represented by a closed circle inside an open circle. Figure 8-1 is a simple activity diagram for the Check Out Asset use-case.

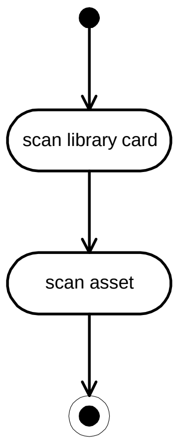

Figure 8-1: Check Out Asset activity diagram

Boxes with rounded corners represent activity states. Each activity state is labeled with a brief description of the activity it represents. The arrows between states, called transitions, represent the shift from one activity state to the next.

CONDITIONALS

A conditional action in an activity diagram is an action that depends on one or more defined requirements. For instance, in the above Check Out Asset use-case, before an asset can be checked out, the system checks the patron's account balance. If the patron's account information indicates an unpaid overdue fine, the system enters an alternative thread of execution in which the library fine is paid. If no overdue fine is due, execution continues along the primary path. Conditionals are represented in activity diagrams with branches and merges.

BRANCHES AND MERGES

Branches represent places where conditionals are used for decisions in the activity flow. When a transition enters a branch, a question is asked and a decision is made to continue the flow in one of two or more possible paths of execution. The paths exiting a branch are called alternative threads because execution continues along only one path. Figure 8-2 below, using the Check Out Asset use-case described above, adds a branch with two alternative threads. A diamond represents the branch. The transitions exiting the branch are labeled to indicate which alternative each represents. The alternative threads converge at a merge, also represented by a diamond. By convention, the primary thread of execution follows a straight path down the page.

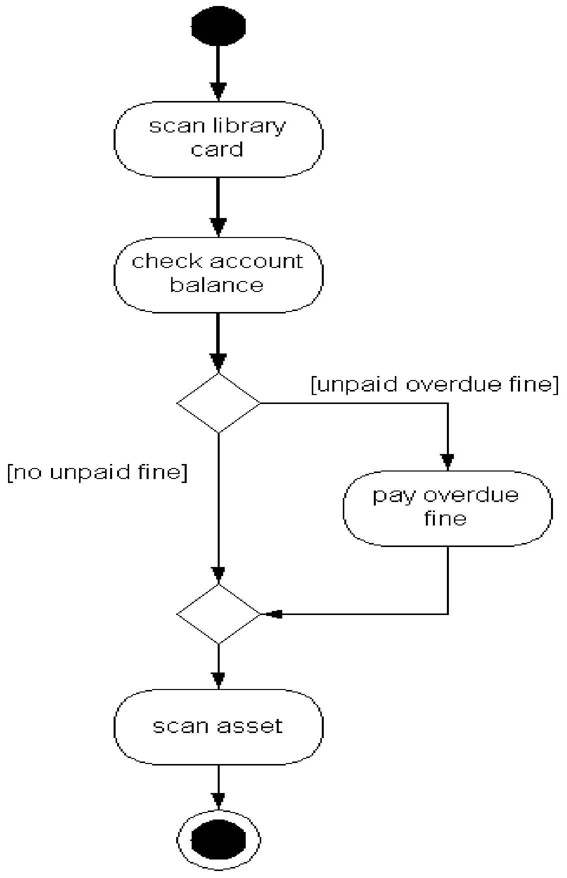

Figure 8-2: Check Out Asset branch and merge

CONCURRENCY

While branches are useful for describing when there are two or more options to take, there are also situations where the action will flow along two or more paths at the same time. This is known as concurrency. For instance, considering the Check Out Asset use-case, when an asset is being checked out, there are two actions that must be performed at the same time: adding that asset to the patron's account information and marking that asset as checked out. These situations are described in Activity Diagrams through forks and joins.

FORKS AND JOINS

Forks and joins distinguish activity diagrams from traditional flow charts. When a transition enters a fork, execution branches off in two or more directions simultaneously. The threads exiting a fork operate concurrently. Threads converge at a join.

Consider the Check Out Asset use-case described above. This sequence can be represented using a fork. Figure 8-3 adds a fork, represented by a horizontal bar, with two concurrent threads. After an asset is scanned, execution continues along each of these threads simultaneously. The threads converge at a join, also represented by a horizontal bar.

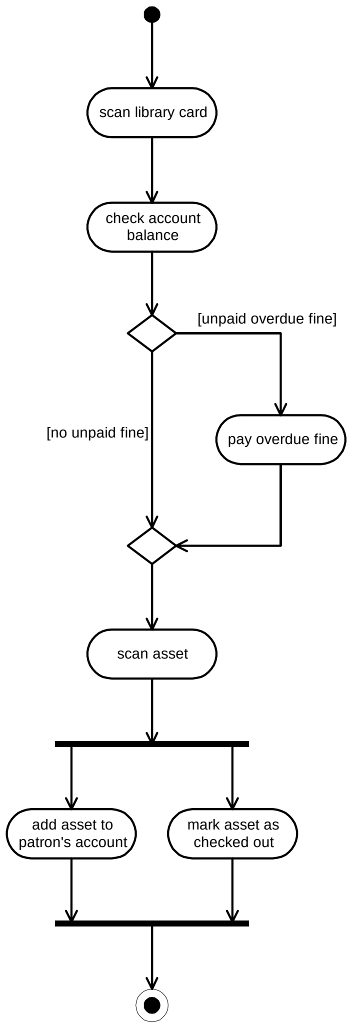

Figure 8-3: Check Out Asset fork and join

LIBRARY SYSTEM ACTIVITY DIAGRAMS

Activity diagrams provide a more complete description of a project's use-cases. Figures 8-4 through 8-12 are activity diagrams for the library system example. As you examine these diagrams, consider how each is used to illustrate the sequence of activities involved in a use-case interaction.

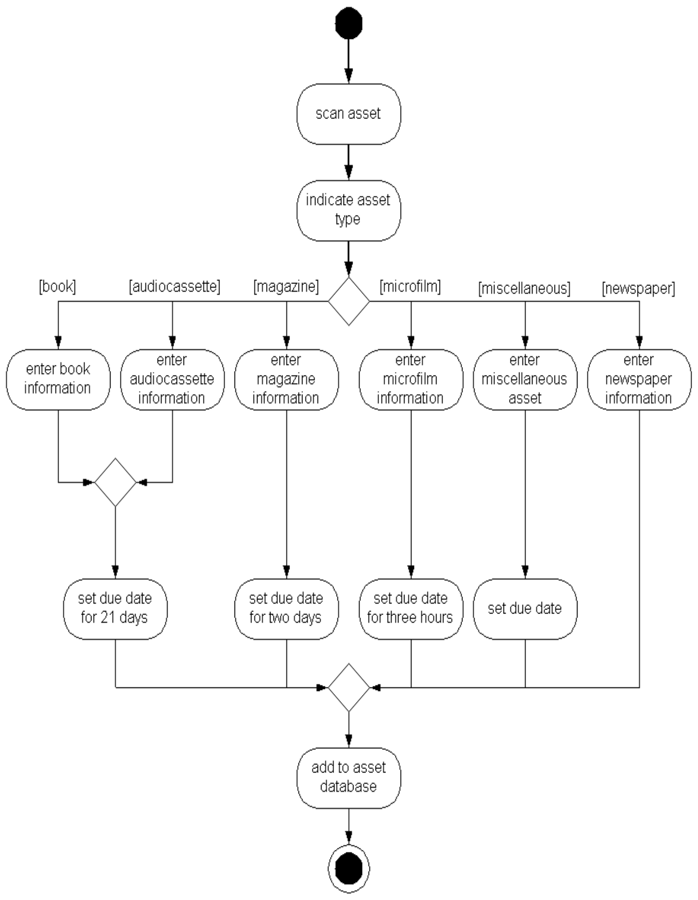
scan asset
indicate asset type
[book]
[audiocassette]
[magazine]
[microfilm]
[miscellaneous]
[newspaper]
enter book information
enter audiocassette information
enter magazine information
enter microfilm information
enter miscellaneous asset
enter newspaper information
set due date for 21 days
set due date for two days
set due date for three hours
set due date
add to asset database

Figure 8-4: Add Asset to Database activity diagram

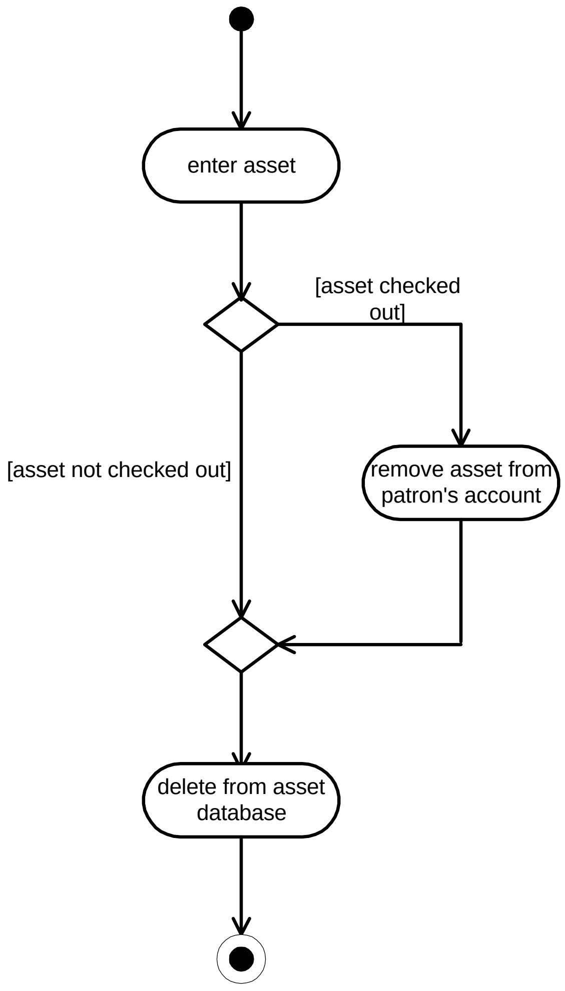
enter asset
[asset checked out]
[asset not checked out]
remove asset from patron's account
delete from asset database

Figure 8-5: Delete Asset from Database activity diagram

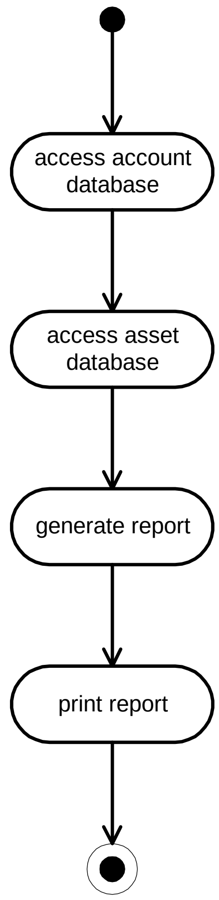

Figure 8-6: Generate Report activity diagram

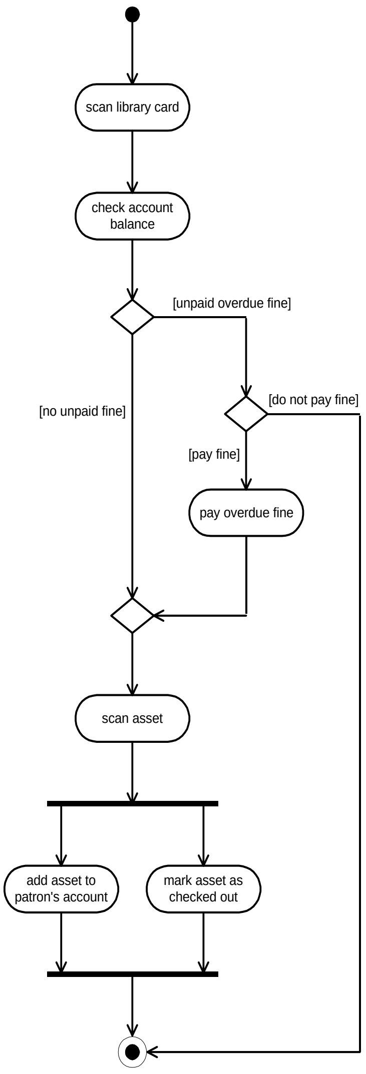

Figure 8-7: Check Out Asset activity diagram

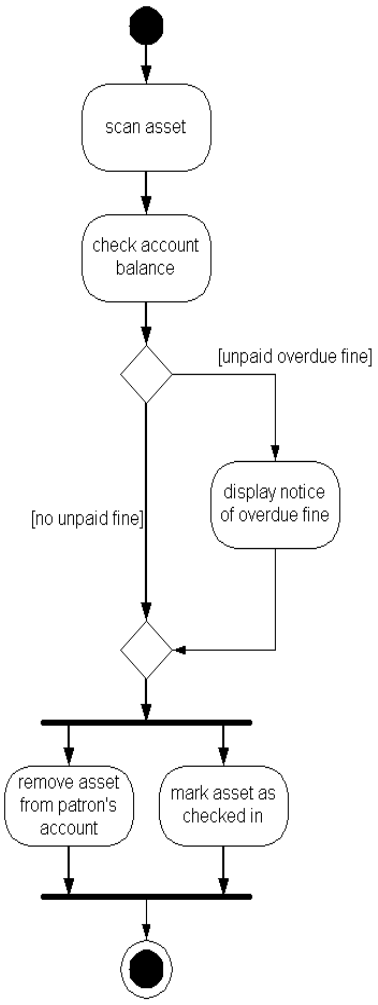

Figure 8-8: Check In Asset activity diagram

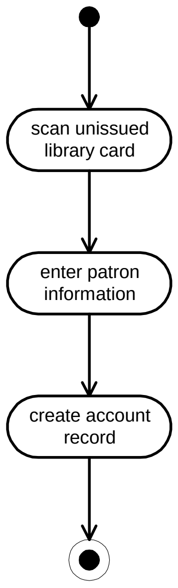

Figure 8-9: Issue Library Card activity diagram

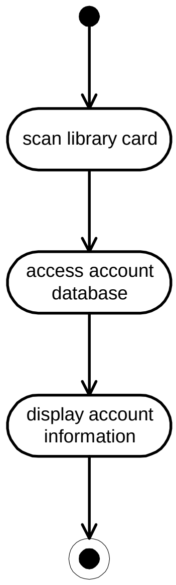

Figure 8-10: Access Account Information activity diagram

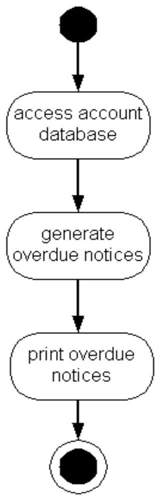

Figure 8-11: Generate Overdue Notices activity diagram

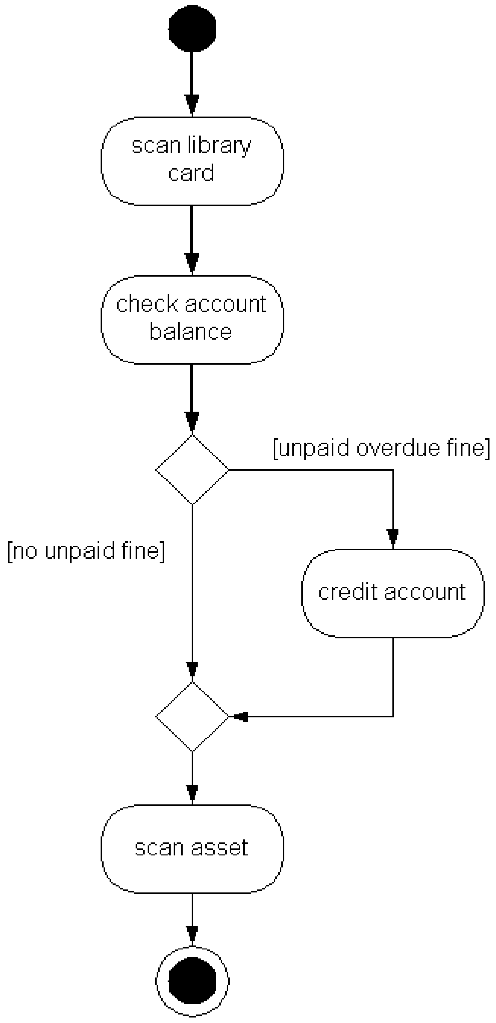

Figure 8-12: Pay Overdue Fine activity diagram

ITERATIONS

While all of the above activity diagrams describe situations which follow a direct path with no stops from beginning to end, there are situations which will require certain actions to be performed more than once before an entire process can be completed. These situations can be described using iterations. An iteration is a step in a process which will repeat until a certain requirement is met.

In the following example, consider a student working on a test. Until all the problems on the test are completed, the student will not be finished working on the test. The loop above the activity state of `Completing Test Questions` indicates that this step will be iterated until all the questions are answered.

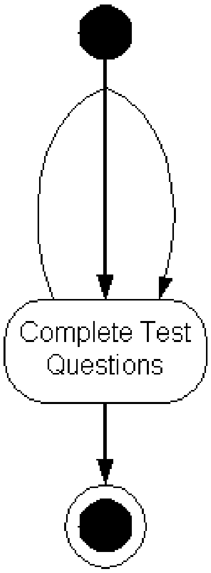

Figure 8-13: Iterations

WHEN THIS IS USEFUL

Activity diagrams are a useful evolution in flowchart design, but they are not always the best way of describing situations. They should be used with care, as with any software design tool. Using an activity diagram when another method of modeling would be more precise or informative can be a large hindrance to efficient software production.

A major advantage of using activity diagrams is the ability to describe parallel behavior, or how behaviors in more than one use-case will interact. But activity diagrams are not designed to describe which specific objects will perform which actions. Rather, an activity diagram can be used to show that the actions taken will influence other actions. Other types of diagrams, such as interaction or state transition diagrams, should be used when attempting to describe objects and methods of messaging used between the objects.

Exercise 8-1: Creating activity diagrams

1. Examine the use-case diagrams you created in the previous chapter's exercise. Identify use-cases that could benefit from further explanation using activity diagrams. Most or all of your top-level use-cases should be included in this group.
2. Using Argo/UML, create an activity diagram for each of the use-cases you identified in Step 1 of this exercise. Refer to Appendix A for assistance using Argo/UML.

SUMMARY

Activity diagrams are similar to flow charts. They describe a process: the order of activities and the branching logic. They can also represent concurrent operations. One common use for activity diagrams is to supplement the description of use-cases. Activity diagrams flow from top to bottom through a series of activity states. Branches are used to represent the logic involved in a process. The paths exiting a branch represent alternative threads of execution. Branches converge at a merge. Forks are used to indicate the beginning of concurrent threads of execution. Forks converge at a join.

POST-TEST QUESTIONS

The answers to these questions are in Appendix A at the end of this manual.

1. Contrast Branches and Forks?

   ....

   ....

2. What are the initial and final states of an Activity diagram represented by?

   ....

   ....