

CHAPTER 8

Use Case Diagrams

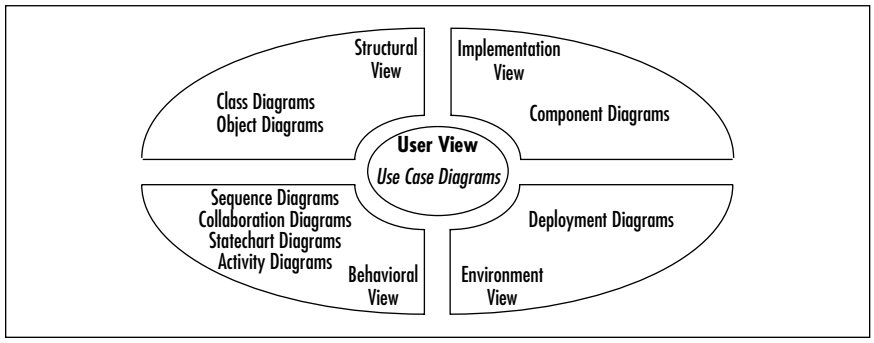


Figure 8-1: User Model View—Use Case Diagrams

Use case diagrams (Figure 8-1) render the user view of a system. These diagrams describe the functionality provided by a system or class to external interactors. These diagrams contain actors, use cases, and their relationships. They may be constructed using the following technique:

1. Model actors.
2. Model use cases. This is the normal behavior of an entity.
3. Model communicates relationships between actors and use cases.
4. Model extends relationships between use cases. This is exceptional or variant behavior of an entity.
5. Model uses relationships between use cases. This is behavior of an entity that is reused by the services it offers.
6. Refine and elaborate as required.

Use case diagrams

- May be contained in a package stereotyped using the “use case model” keyword. This package represents the specification of requirements of a system.
- May correlate to a package stereotyped using the “top level package” keyword. This package represents the specification of the implementation of a system that satisfies the requirements.
- May be part of a package stereotyped using the “system” keyword that contains a “use case model” package, “top level package” package, and a set of classes organized around analysis and design models in a package hierarchy.
- May have the various expressions (conditions, etc.) they utilize be expressed using pseudocode or another language.

Actors

Actors (Figure 8-2) are classes that define roles that objects external to a system may play. They are used to model users outside of a system that interact directly with the system as part of coherent work units. This includes human users and other systems. Actors

- Are characterized by their external view rather than their internal structure.
- Participate in interactions involving message exchanges and actions with systems.

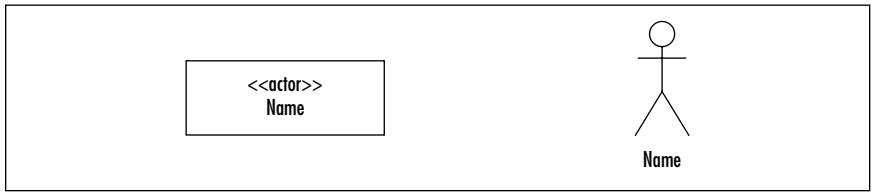


Figure 8-2: Actors

- Are denoted as stereotyped class rectangles or as stick person icons.
- Have goals to be achieved by interacting with systems.
- Have instances that are objects playing the role of an actor.
- Have one role for each use case with which they interact, that is, one role per communicates relationship.
- Must be stereotyped using the “actor” keyword. This is usually used for system interactors and nonhuman systems.
- Must have a name or identifier string that represents the name of the actor.
- May be denoted using the standard stereotype icon (stick person). This is usually used for human interactors.
- May be played by a single object; that is, an object may play several roles simultaneously.
- May have generalization relationships with other actors. That is, an actor may inherit the characteristics of a more general actor.

- May not have aggregation relationships with other classifiers. That is, an actor may not contain any other classes.
- May have operations.
- May have interfaces that define how other actors may interact with them.

Use Cases

Use cases (Figure 8-3) are classes that define units of functionality or behavior provided by a system. They specify the external requirements of the system and the functionality offered by the system. The set of use cases may be enclosed by a system boundary or rectangle labeled with the system name. The system container is an actor. Use cases are used to model work units that the enclosing system provides as services to outside interactors. The system is enclosed by a boundary and performs the functionality for actors. Use cases

- Are *specified* by sequence diagrams (Chapter 9) representing the external interaction sequences among the system and its actors. These interaction sequences represent how a system provides functionality or services to its interactors.
- Are *realized,* or implemented, by collaboration diagrams (Chapter 10) representing the internal refinement of the services provided by a system to its actors.



Figure 8-3: Use Cases

- Are denoted as ellipses or ovals.
- Yield results of value to at least one of their actors; however, there may be other actors who simply participate and gain no value from their participation. When use cases are not complete specifications and are just fragments of functionality, they may be at the incorrect level of abstraction.
- Have instances that are scenarios or single paths of activity through use cases. Use cases are reified through scenarios that may be depicted as sequence diagrams (Chapter 9) or collaboration diagrams (Chapter 10).
- Must have a name or identifier string that represents the name of the use case.
- May not have aggregation relationships with other classifiers.
- May have operations and attributes.
- May have interfaces that define and describe how actors may interact with them.
- May be described in plain text or using behavioral view diagrams.

- May have extension points, which
  - Are strings representing points within a use case at which variations in the action sequence may be inserted.
  - Must have unique names in a use case.
- May be abstract. Abstract use cases are never used by themselves; that is, they do not have any communicates relationships with actors, but they may have uses or extends relationships with other use cases.

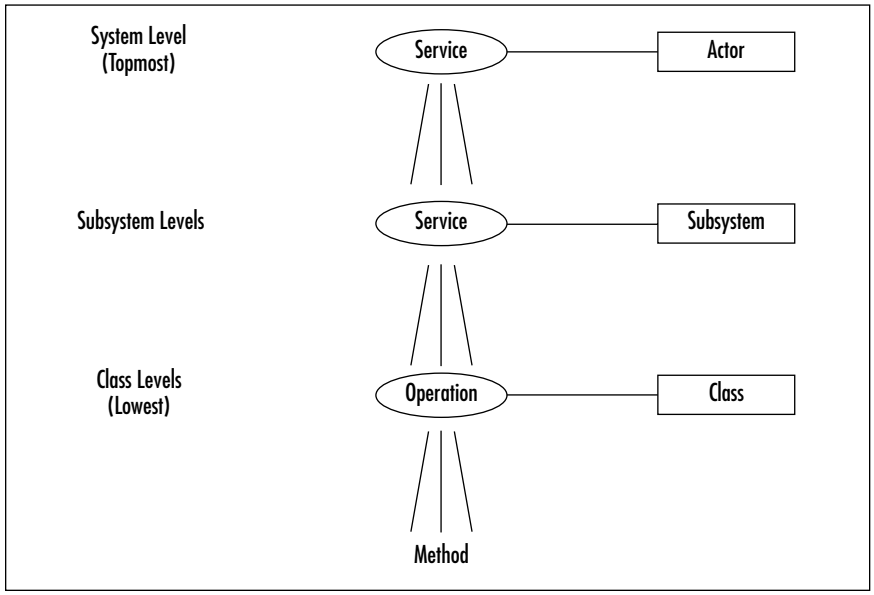


Figure 8-4: Use Case Organization

- Are organized hierarchically (Figure 8-4). The hierarchy organizes the overall functionality provided by a system. Within such a hierarchy, use cases have the following characteristics:
  - Use cases specify the services offered by their containers.
  - Use cases are refined into a set of smaller use cases. The use cases of a container are superordinate to refining use cases; the refining use cases are subordinate to the use cases of the whole.
  - Subordinate use cases collaborate to perform superordinate use cases.
  - Actors of superordinate use cases may appear as actors of subordinate use cases.
  - Interfaces of superordinate use cases may appear as interfaces of subordinate use cases.
  - Cooperating use cases are actors of one another.
  - The functionality of superordinate use cases is traceable to their subordinate use cases. That is, the functionality provided by a superordinate use

case is a composite of the functionality provided by each of its subordinate use cases.

  - Interfaces of superordinate use cases are traceable to interfaces of subordinate use cases that communicate with the same actors of the superordinate use cases. That is, the interfaces of a superordinate use case are the summation of the interfaces provided by its subordinate use cases.

- May be used by a system. These use cases
  - Specify the complete services offered by the system. This is the topmost level in a use case diagram and is known as the *system level*.
  - Are realized by interaction sequences or collaborations among multiple elements contained within the system.
  - Have external interactors that are actors or other systems.
  - Use signals to interact with their external interactors.
- May be used by a subsystem within a hierarchy. These use cases
  - Specify the complete services offered by the subsystem. Any number of levels involving subsystems may be utilized. These are known as *subsystem levels*.
  - Are realized by a single element or by interaction sequences or collaborations among multiple elements contained within the subsystem.
  - Have external interactors that are other subsystems and higher-level interactors.
  - Use signals or other communication mechanisms to interact with their external interactors.
  - Realize the next higher level of use cases.
  - Specify the next lower level of use cases.
- May be used by a class within a hierarchy. These use cases
  - Specify functional fragments or operations offered by the class. Any number of levels involving classes may be utilized. These are known as *class levels*.
  - Are realized by a single element or by interaction sequences or collaborations among multiple elements contained within the class.
  - Have external interactors that are other classes and higher-level interactors.
  - Use signals or other communication mechanisms to interact with their external interactors.
  - Realize the next higher level of use cases.
  - Specify the next lower level of use cases.
- May be used at the lowest level of a hierarchy. These use cases
  - Are methods.
  - Realize the next higher level of use cases.

Communicates Relationships

Communicates relationships (Figure 8-5) are associations between actors and use cases. They are used to model communications between actors and use cases in which an actor participates, communicates with, or takes part in a use case.

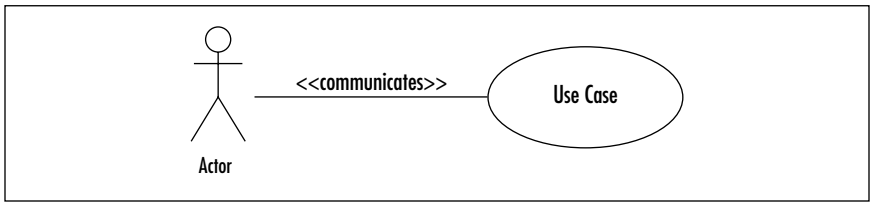


Figure 8-5: Communicates Relationship

Communicates relationships

- Are denoted as solid associations.
- Are the only relationships permissible between actors and use cases.
- May be stereotyped using the "communicates" keyword.
- May have an arrow to indicate the actor's behavior within the use case.
  - If the actor initiates the use case, the communication association has an arrow pointing to the use case.
  - If the actor utilizes the services provided by the use case, the communication association has an arrow pointing to the actor.
  - An association may have arrows in both directions.
- At the topmost level within a use case hierarchy, involve signals or uninterruptible atomic transactions.
- At lower levels within a use case hierarchy, involve various communication mechanisms.
- Allow a single actor to participate in any number of use cases.
- Allow a single use case to have any number of participating actors.

Extends Relationships

Extends relationships (Figure 8-6) are generalizations between use cases. They are used to model relationships between use cases in which a base use case instance may include the behavior specified by an extending use case, subject to conditions specified in the extension. Extends relationships

- Are used to capture exceptional behavior (or variations of normal behavior).
- Allow an extending use case to continue the activity sequence of a base use case when the extension point is reached in the base use case and the extension condition is fulfilled. Upon completion of the extension activity sequence, the original use case continues.
- Are denoted as generalization arrows.

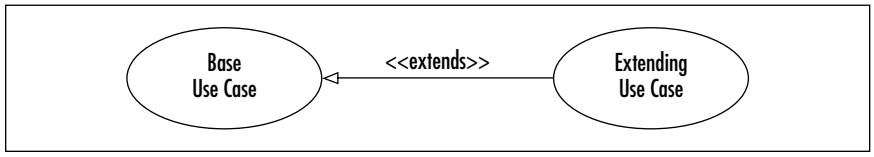


Figure 8-6: Extends Relationship

- Must be stereotyped using the "extends" keyword.
- May exist between use cases within multiple systems, but only if the system containing the extending use case specializes, via a generalization relationship, a system containing the base use case.
- Must have a condition determining when the extending use case will be inserted.
- May reference an extension point in the base use case determining where the extending use case may be inserted.

  Extending use cases may insert their parts (subordinate use cases) at different extension points within the base use case. Each part may have its own condition; however, if only one condition is used for the overall extending use case, all parts are inserted in the base use case if the condition is satisfied.

Uses Relationships

Uses relationships (Figure 8-7) are generalizations between use cases. They are used to model relationships between use cases in which a base use case instance will also include the behavior specified by a common use case.



Figure 8-7: Uses Relationship

Uses relationships

- Are used to share common behavior among use cases.
- Are denoted as generalization arrows.
- Must be stereotyped using the "uses" keyword.
- May exist between use cases within multiple systems, but only if the system containing the base use case specializes, via a generalization relationship, a system containing the common use case.

  Base use cases may interleave all their associated common use cases together with new pieces of behavior.

,ch08.18171 Page 166 Wednesday, May 9, 2001 4:03 PM