Part I

Modeling Artifacts and Bridge Process

This part introduces modeling artifacts that are used to express architectures in terms of uses cases and objects. The second chapter introduces the ***bridge*** *process.*

Chapter 1

Modeling Concepts, Artifacts, and Relations

This chapter introduces modeling concepts, artifacts, relations, and the corresponding graphical representations. It relies on examples expressed in UML graphical notation. Most of the examples introduced are built upon in subsequent chapters to give thorough examples.

1. ACTOR

An ***actor*** is something or someone that interacts with the target system to produce an observable result. An actor falls on the boundary of the system. Actors model the roles of real users using the system for different purposes or systems that interact with the software system, such as a border system that receives input from the main system. A system may have one or more actors. A real user that has different roles with respect to uses of a system could be modeled by more than one actor.

Classes are the corresponding implementation concepts of actors. An actor class may have features and attributes like any other class.

Following is the UML stick-person notation for representing actors that model real users. A classlike representation is used to model nonreal users. Each actor is given a name that is selected to reflect the role that an actor models.

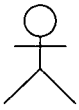


Figure 1-1 shows three actors: *Broker*, *Buyer*, and *Seller* of a real estate system.

Broker is an actor who is the owner of the system and can use the system to update data on real estate property. *Buyer* and *Seller* are actors that can use the system to browse through the available property list and place purchase orders or sale requests. Note that *Broker*, *Buyer*, and *Seller* could be roles for a single real user who is the owner of the real estate software. The same actors may represent different real users for an Internet real estate system where buyers and sellers can interact with the software through the web or other means to place a purchase order or to post a real estate property for sale.

Figure 1-2 shows three actors, *Warehouse Employee*, *Customer*, and *Vendor*, of a software solution for a company that resells goods. It obtains the products from vendors and resells to customers.

Warehouse Employee is the active user of the system who can update data on goods for all warehouses. *Vendors* are passive actors who receive requests from *Customer* actors through the system and respond to these requests when activated by the *Warehouse Employee*.

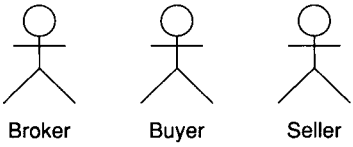


***Figure 1-1.* Actors of a real estate system**

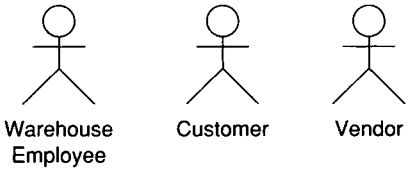


***Figure 1-2.* Actors of a reseller company**

Actor Categories

In general, actors fall into two categories: ***Active*** actors, who initiate interactions with a system, such as persons who use the system to perform an activity, and ***passive*** actors, who receive requests or input from a system and are activated by these to carry an activity.

For example, *Warehouse Employee* actor of Figure 1-2 is an active actor because he issues requests to a system. *Vendor* is a passive actor because he responds to requests from the system when processed by *Warehouse Employee* actor. Figure 1-3 shows two active actors and one passive actor: *Employee*, *Customer*, and *Accounting System* of an online rent-a-car software system.

The system permits users to make online reservations of cars. Rental charges are computed by a boundary accounting system. Both *Employee* and *Customer* are active actors. *Employee* is an actor in charge of the reservation system and *Customer* is an actor that uses the system to effect a reservation. *Employee* and *Customer* are two different actors because they stand for two different roles. The accounting system is a passive actor that receives the details of the reservation and computes the corresponding rental charges. Note the stick-person representation of real users versus the classlike representation of a nonreal actor.

Generalization among Actors

Actors may be related by a ***generalization relationship***. Specialization relation is the opposite of generalization relation, and it is similar to inheritance relation among classes:

An actor A is a generalization of an actor B (B inherits A) whenever B and A are related by the is-a relation as follows: B is-a A. This permits B to assume the same roles modeled by actor A, whereas actor A need not necessarily play the same roles as B. A is the parent actor, and B is the descendant actor.

Generalization is a one-way transitive relation. Consequently, if actor A is a generalization of an actor B and B is a generalization of an actor C, then A is a generalization of actor C and consequently actor C can assume the same roles modeled by A.

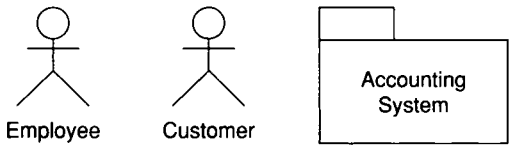
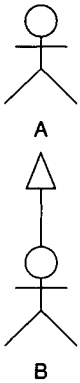


Figure 1-3. **Actors of a rent-a-car system**

The following is a template of the generalization relation between two actors.



Actor's generalization

Figure 1-4 shows an application of generalization relation to *Customer* and *Employee* actors of a rent-a-car system. An employee may be a customer and use the system to order a car but a customer is not necessarily an employee.

Figure 1-5 shows another example of a generalization relation where one actor could have one or more descendants. Actors *System Manager* and *Information Security Manager* are descendants of actor *IT Manager*; both are IT managers but an IT manager is not necessarily a specialized manager.

An actor could be a parent of more than one actor – each of its descendants could assume roles modeled by the parent actor.

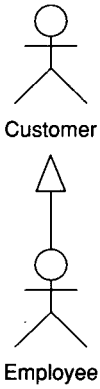


***Figure 1-4.* An employee as a customer**

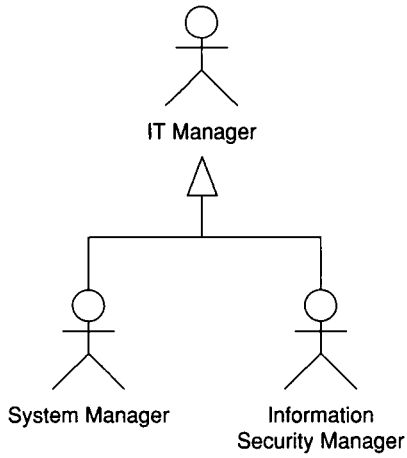
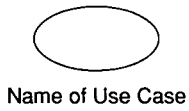


***Figure 1-5.* Managers of an IT department**

2. USE CASE

A ***use case*** is one instance of how an actor would use a software system to activate a business function that is a service offered by the system and to produce a result. With use cases it is possible to specify all services offered to users by the system when use cases are related to actors these directly specify functionality of the system.

A use case is modeled with an oval and a descriptive name string that describes its functions. The following is the notation for modeling a ***use case***.



Use case

Use cases typically stand for verb phrases that can be derived from user requirements and from possible uses of the system by actors.

Figure 1-6 shows three use cases of a real estate property: *Browse*, *Sell a Property*, and *Buy a Property*. Each of these use cases indicates a single service provided by the system. The three reflect the main functionality of the real estate property system. *Browse* permits a user to browse existing data of real estate, *Sell a Property* permits a user to add real estate to the existing data, and *Buy a Property* permits a user to place a purchase order.

Figure 1-7 shows an example of four use cases of an online chat system: *Add User, Remove User, Receive Message,* and *Send Message*. The system is designed so that it allows users to start chatting from within a chat room. A user can enter a

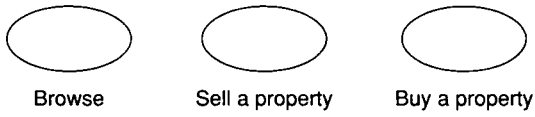


***Figure 1-6.* Use cases of a real estate property**

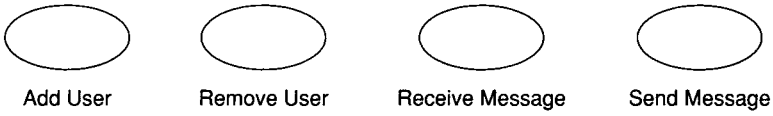


***Figure 1-7.* Use cases of an online chat system**

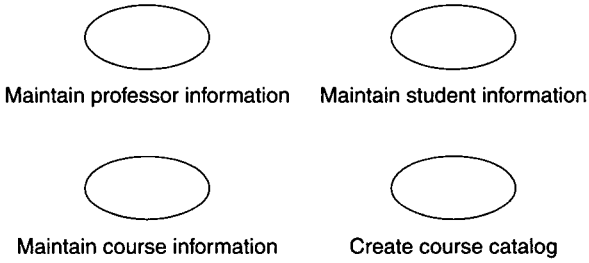


***Figure 1-8.* Use cases of a university registration system**

session by logging into the chat room; chatting involves sending and receiving messages from other logged in users. A user can terminate a chat session by removing himself from the chat room. Each of the four use cases models one service that could be utilized by users. The *Add User* and *Remove User* use cases allow a chat user to add or remove himself from the chat room. The *Receive Message* and *Send Message* use cases allow a chat client to communicate with other chat clients.

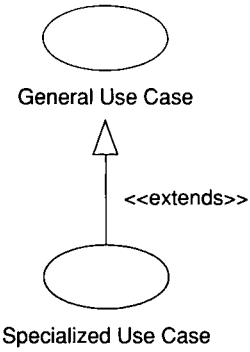
Figure 1-8 shows another example of use cases of a university registration system that permits registering students in courses taught by professors. The *Maintain Course Information* and *Create Course Catalog* use cases model services offered by the system to the registrar office to update a course content or to add a new course in the system. The *Maintain Professor Information* and *Maintain Student Information* use cases model services of the system that permit the registrar office to update the information regarding university's professors or students.

Use cases could be used to model general services of a system or to model simple atomic actions. It all depends on the intended level of granularity to be reached by a system analyst during the analysis stage.

Extend Relationship

Use cases may be related by an ***extend relation***. An extend relationship is a generalization relationship where one use case extends another use case by adding actions to it.

The following is a template of the extend relation between use cases.



Extends relationship

In the boat-purchase example (Figure 1-9), an *extends* relation is applied to two use cases, *Signing Boat Purchase Contract* and *Signing Insurance Policy*, because signing an insurance policy is a part of, and typically precedes, signing a purchase contract.

In Figure 1-10, the use case *Search* extends *Browse* because every search, when it commences, requires a browsing activity.

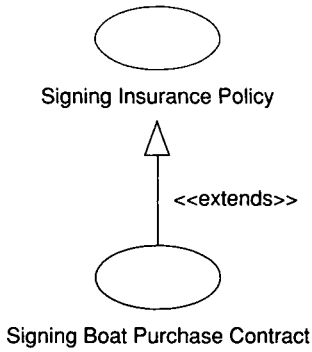


Figure 1-9. **Boat purchase**

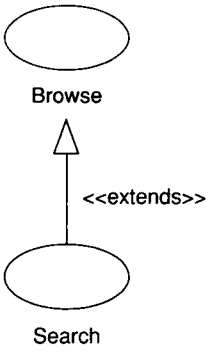
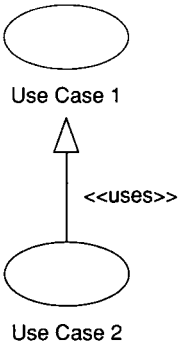


***Figure 1-10.* Search activity**

Uses Relationship

Use cases may be related by a ***uses relation***. ***Uses*** is a generalization relationship where one use case uses another use case indicating that, as part of the specialized use case, the behavior of the general use case will also be included.

The following is a template of uses relation between use cases.



Uses relation

The *uses* relation is different from the *extends* relation. In fact, the *extends* relationship involves inheritance because the specialized use case is a subset of the general one. In the *uses* relation, the two use cases might not have any kind of inheritance but one still uses the other for a specific purpose.

In the bookstore inventory system, we have the *Locates Book* use case that uses the *Gives Price* use case. In this case the ***uses*** relation is more appropriate than the ***extends*** relation because whenever a book is located it has to return the price of the item and, hence, it uses the use case *Gives Price*. This is modeled in Figure 1-11.

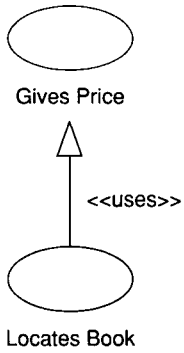


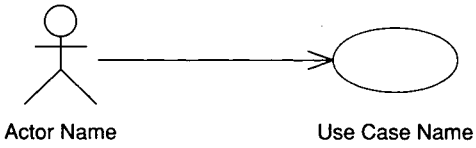
Figure 1-11. **Bookstore inventory system**

3. USE CASE MODEL

A ***use case model*** consists of actors and use cases. It captures services offered by a system and users of the system in terms of actors, use cases, and the *communicate* relationship.

Communicate Relation

The following is a template of a communicate relation (also referred to as an association) between an actor and a single use case. The direction of the arrow indicates the direction of flow of information and direction of interaction. An association could also be outgoing from the use case to the actor to model the response by the system to actors.



Actor associated with a use case

The relation between a use case and an actor could take several forms: one to one, one to many, or many to many. That is, an actor could be associated with a single use case; in this case, instances of the actor can communicate with the system to get the single service provided by the use case. In a one-to-many relation an actor could communicate with the system to obtain all the services that are captured by the use cases with which the actor is associated. A many-to-many relation permits more than one actor and, consequently, their instances to get the same services from a system.