Use case diagrams depict:

- **Use cases.** A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

- **Actors.** An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.

- **Associations.** Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case. The arrowheads are typically confused with data flow and as a result I avoid their use.

- **System boundary boxes (optional).** You can draw a rectangle around the use cases, called the system boundary box, to indicates the scope of your system. Anything within the box represents functionality that is in scope and anything outside the box is not. System boundary boxes are rarely used, although on occasion I have used them to identify which use cases will be delivered in each major release of a system.

- **Packages (optional).** Packages are UML constructs that enable you to organize model elements (such as use cases) into groups. Packages are depicted as file folders and can be used on any of the UML diagrams, including both use case diagrams and class diagrams. I use packages only when my diagrams become unwieldy, which generally implies they cannot be printed on a single page, to organize a large diagram into smaller ones.

Figure 1. System use case diagram.
In the example depicted in Figure 1 students are enrolling in courses with the potential help of registrars. Professors input the marks students earn on assignments and registrars authorize the distribution of transcripts (report cards) to students. Note how for some use cases there is more than one actor involved. Moreover, note how some associations have arrowheads; any given use case association will have a zero or one arrowhead. The association between Student and Enroll in Seminar (Figure 4) indicates this use case is initially invoked by a student and not by a registrar (the Registrar actor is also involved with this use case). Understanding that associations don’t represent flows of information is important; they merely indicate an actor is somehow involved with a use case. Information is flowing back and forth between the actor and the use case, for example, students would need to indicate which seminars they want to enroll in and the system would need to indicate to the students whether they have been enrolled. However, use case diagrams don’t model this sort of information. Information flow can be modeled using UML activity diagrams. The line between the Enroll in Seminar use case and the Registrar actor has no arrowhead, indicating it is not clear how the interaction between the system and registrars start. Perhaps a registrar may notice a student needs help and offers assistance, whereas other times, the student may request help from the registrar, important information that would be documented in the description of the use case. Actors are always involved with at least one use case and are always drawn on the outside edges of a use case diagram.

Reuse Opportunities

Figure 4 shows the three types of relationships between use cases -- extends, includes, and inheritance -- as well as inheritance between actors. I like to think of extend relationships as the equivalent of a "hardware interrupt" because you don't know when or if the extending use case will be invoked (perhaps a better way to look at this is extending use cases are conditional). Include relationships as the equivalent of a procedure call. Inheritance is applied in the same way as you would on UML class diagrams -- to model specialization of use cases or actors in this case.

Figure 4. Use case reuse.

Creating Use Case Diagrams

I like to start by identifying as many actors as possible. You should ask how the actors interact with the system to identify an initial set of use cases. Then, on the diagram, you connect the actors with the use cases with which they are involved. If an actor supplies information, initiates the use case, or receives any information as a result of the use case, then there should be an association between them. I generally don’t include arrowheads on the association lines because my experience is that people confuse them for indications of information flow, not initial invocation. As I begin to notice similarities between use cases, or between actors, I start modeling the appropriate relationships between them.

The preceding paragraph describes my general use case modeling style, an “actors first” approach. Others like to start by identifying one actor and the use cases that they’re involved with first and then evolve the model from there. Both approaches work. The important point is that different people take different approaches, so you need to be flexible.