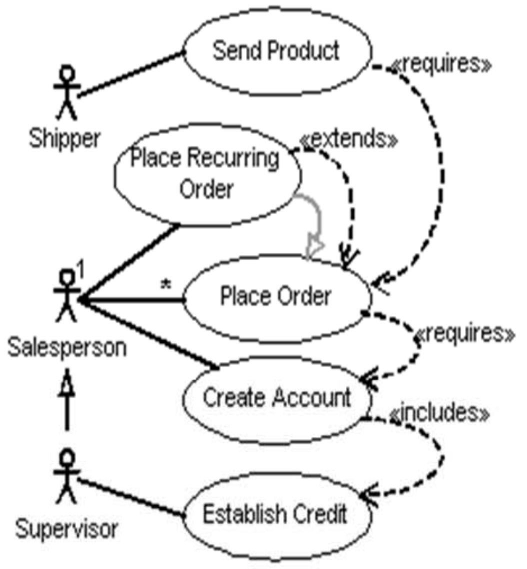

Specifies participants in a use case and the relationships between use cases.

- The stick-figure represents a role taken on by some actor (sometimes called simply "actor," but it's really a role).
- A line connects the actor/role to the use case in which it participates. You may use cardinality. (A Salesperson places many orders.)
- An is-specialization-of/generalizes relationship between actor/roles (denoted by ◁—) indicates additional responsibilities. (A *Supervisor* has all the responsibilities of a *Salesperson*, but can also establish credit. A *Supervisor* can create an account, for example.)
- Dotted lines denote use-case dependencies. Common dependencies are:

«equivalent» Equivalent use cases have identical activities and identical flow, but end users think of them as different. ("Deposit" and "Withdrawal" might have identical activities, though the objects involved might be different.)

«extends» When *extension* extends *base*, all the activities of the *base* use case are also performed in the *extension* use case, but the *extension* use case adds additional activities to —or slightly modifies existing activities of—the *base* use case. (To place a recurring order, you must perform all the activities of placing an order plus set up the recurrence.)

If a set of activities occur in several use cases, it's reasonable to "normalize" these common activities out into a *base* use case, and then extend it as necessary.

Holub Extension: This relationship is really a form of derivation, so I use the derivation arrow (◁—) instead of a dashed line. As in a class diagram, the arrow points from the *extension* to the *base* use case.

«includes» A subcase. If *case* includes *subcase*, then the activities of *subcase* are performed one or more times in the course of performing *case*. (An "Authenticate" subcase may be included in several larger use cases, for example.) The subcase is usually represented in the using use case as a single box marked with the subcase name and the stereotype **«use case»**.

«requires»
«follows» If *follower* requires *leader*, then *leader* must be completed before you can execute the *follower* use case. (You must create an account before you can place an order.)

«resembles» Two use cases are very similar, but do have different activities.

Actors/roles are mostly uninteresting to programmers. The dependencies are valuable in determining which use case to implement first. (I often implement the use cases that have the most incoming arrows first, since other use cases depend on them.)