UML Specification by Example: Use Cases

- To walk you through data modeling, we will specify use case and data models for a particular database application using UML
- We will also compare the UML data model with its counterpart in E-R
- Refer to SKS Chapter 6 for E-R details
- Finally — there is a real need for this application! If you're interested, talk to me if you want *this* to be your database project

- We start with an overall write-up of the desired application — while it may not contain every painstaking detail, it should give you a decent idea of what the application should do:

We would like to create a student information and document management system, to augment the student records that are already maintained by the university. With this system, a user can maintain a list of student records. Linked to each student record would be a set of documents. Each document would have a timestamp, and can be bound to any number of keywords, which are also defined in a separate list. The actual document files (such as PDF) can be uploaded to the system and stored on a server; they can then be retrieved by (at least) student, date, and/or keyword. The keyword list can be added to as needed, and functions for adding new students and maintaining existing student data would also be required. The preferred end-user interface would be the Web, although the overall architecture may also accommodate other UIs such as Swing.

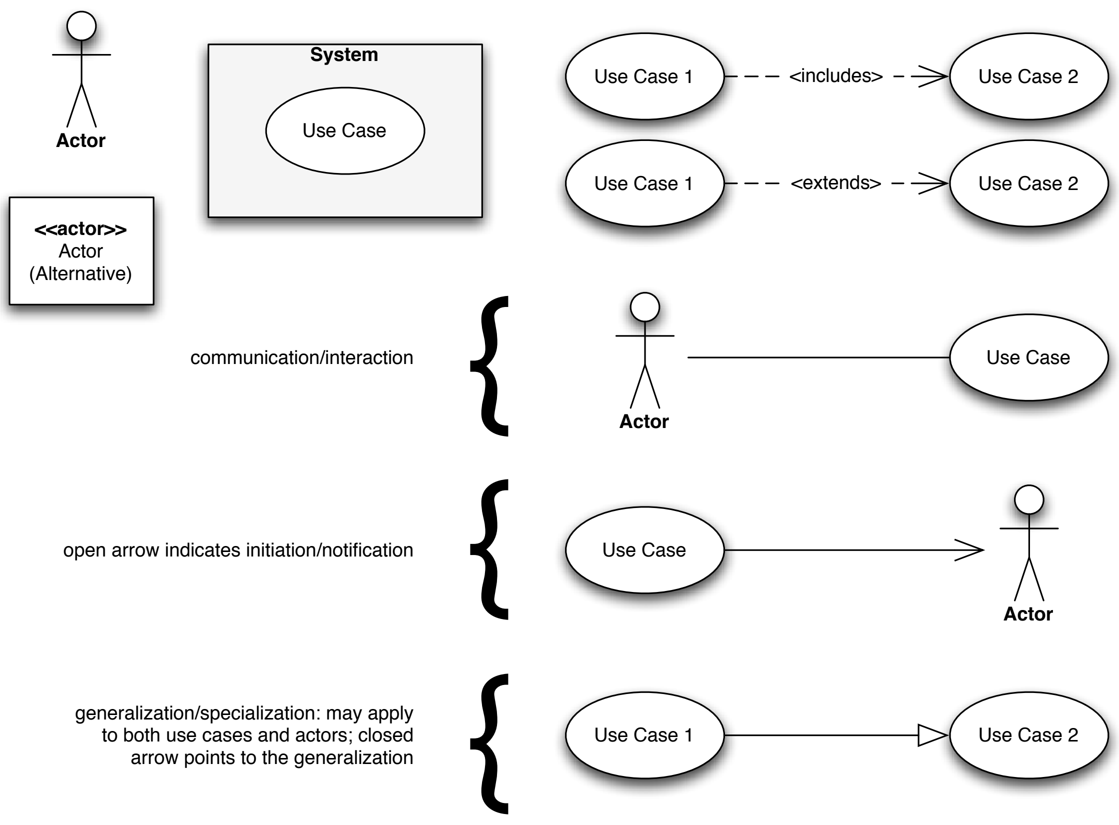
Use Case Modeling

From Alhir, *Learning UML*, O'Reilly 2003:

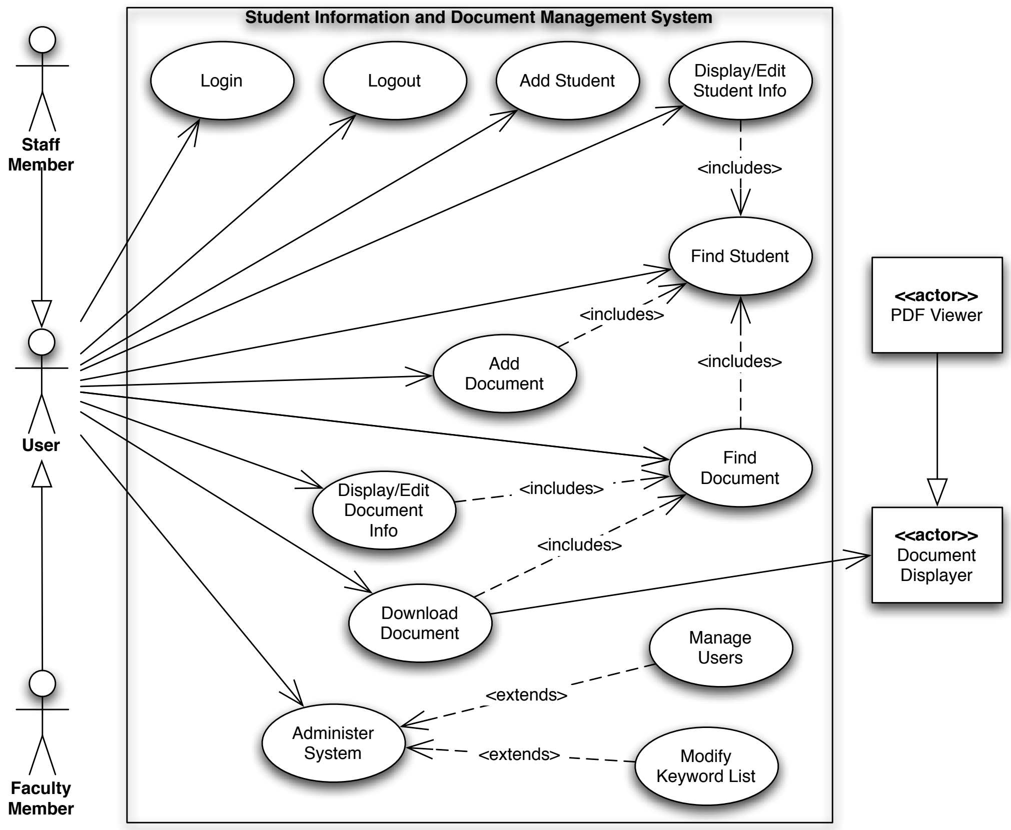
- *System* — what you are building; use case modeling is helps to determine the system's *boundaries*
- *Use case* — a functional requirement that is described from the perspective of the users of a system; an overall *use case model* an be thought of as a "table of contents" to the functional requirements of a system
- *Actor* — a human user or external system with which your system interacts

Use Case Diagram Notation

- Doesn't have to be a fancy diagram — it is just as useful as a straight-up text writeup
- Depending on the necessary level of detail, you'll probably end up writing out some text anyway (detailed steps, use case prerequisites, possible outcomes, exceptions, etc.)
- That said, UML does provide a consistent notation for creating use case diagrams at the top level


Actor
System
Use Case
Use Case 1
<includes>
Use Case 2
Use Case 1
<extends>
Use Case 2
<<actor>>
Actor
(Alternative)
communication/interaction
Actor
Use Case
open arrow indicates initiation/notification
Use Case
Actor
generalization/specialization: may apply to both use cases and actors; closed arrow points to the generalization
Use Case 1
Use Case 2

- Use case 1 *includes* use case 2 if use case 1 "calls" use case 2 within its own flow
- Use case 1 *extends* use case 2 if use case 1 augments the activities in use case 2
- Actors involved in a use case are indicated by a solid line between the actor and the use case, indicating communication or interaction
- A use case model explicitly shows *initiation* — who triggers what — by placing an open arrow in the direction of the "triggeree"
- Use cases and actors may participate in *generalization* and *specialization* — equivalent to subclassing in object-oriented systems



- As with most elements of requirements analysis, use case modeling is iterative — give it a shot, check it against the users, tweak, then check again
- When developers feel that they have enough information to build a system *and* target users feel that the list of use cases will fulfill their needs, some implementation work may occur
- Developers and target users interact throughout the process, as a number of things may happen, such as:
  - New information may emerge
  - Missing information may be discovered
  - Errors/misunderstandings are found