Outline

- What is UML and why we use UML?
- How to use UML diagrams to design software system?
- What UML Modeling tools we use today?

What is UML and Why we use UML?

- UML → “Unified Modeling Language”
- Language: express idea, not a methodology
- Modeling: Describing a software system at a high level of abstraction
- Unified: UML has become a world standard

  Object Management Group (OMG): www.omg.org

What is UML and Why we use UML?

- More description about UML:

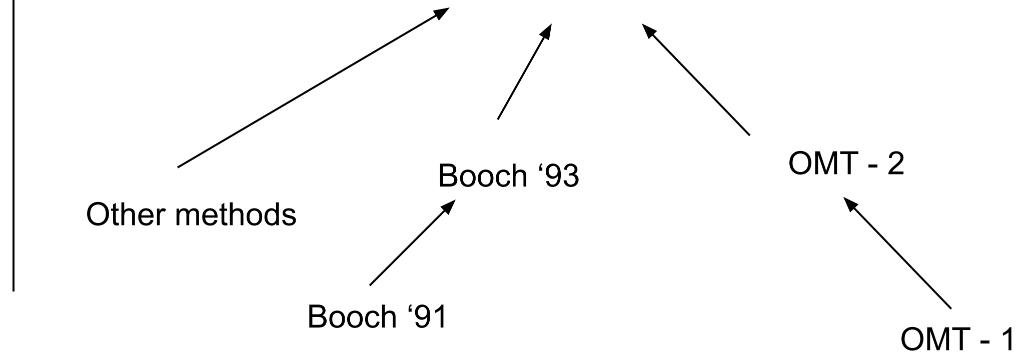
- It is a industry-standard graphical language for specifying, visualizing, constructing, and documenting the artifacts of software systems
- The UML uses mostly graphical notations to express the OO analysis and design of software projects.
- Simplifies the complex process of software design

What is UML and Why we use UML?

- Why we use UML?
  - Use graphical notation: more clearly than natural language (imprecise) and code (too detailed).
  - Help acquire an overall view of a system.
  - UML is *not* dependent on any one language or technology.
  - UML moves us from fragmentation to standardization.

What is UML and Why we use UML?

Year	Version
2003:	UML 2.0
2001:	UML 1.4
1999:	UML 1.3
1997:	UML 1.0, 1.1
1996:	UML 0.9 & 0.91
1995:	Unified Method 0.8



How to use UML diagrams to design software system?

- Types of UML Diagrams:
  - Use Case Diagram
  - Class Diagram
  - Sequence Diagram
  - Collaboration Diagram
  - State Diagram

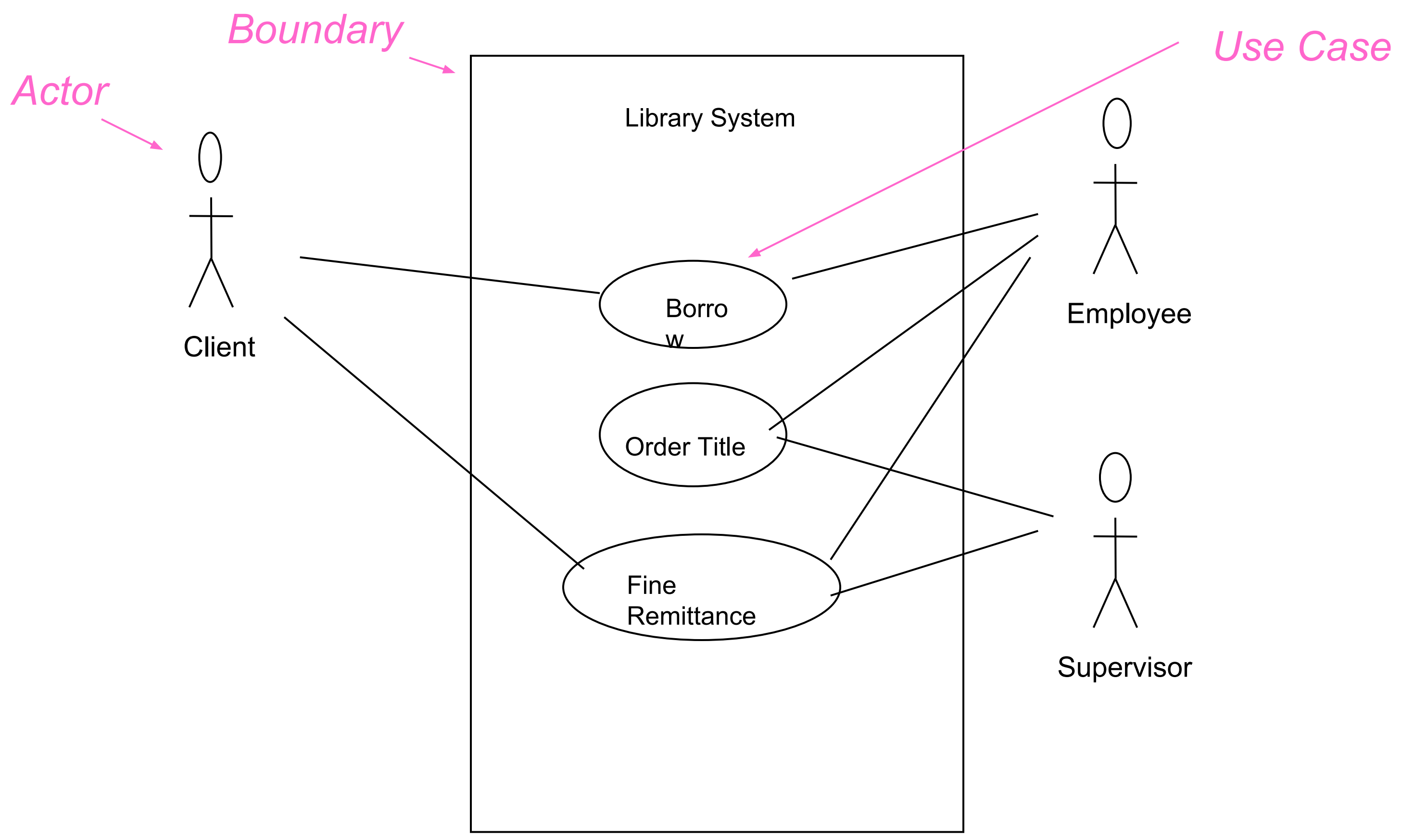
This is only a subset of diagrams … but are most widely used

Use-Case Diagrams

- A use-case diagram is a set of use cases
- A use case is a model of the interaction between
  - External users of a software product (actors) and
  - The software product itself
  - More precisely, an actor is a user playing a specific role

- describing a set of user **scenarios**
- capturing user requirements
- **contract** between end user and software developers

Use-Case Diagrams

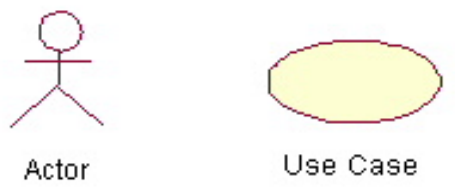


Use-Case Diagrams

- **Actors:** A role that a user plays with respect to the system, including human users and other systems. e.g., inanimate physical objects (e.g. robot); an external system that needs some information from the current system.

- **Use case:** A set of scenarios that describing an interaction between a user and a system, including alternatives.

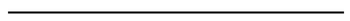
- **System boundary**: rectangle diagram representing the boundary between the actors and the system.



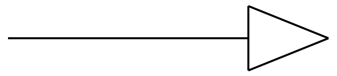
Use-Case Diagrams

- <u>Association:</u>

  communication between an actor and a use case; Represented by a solid line.

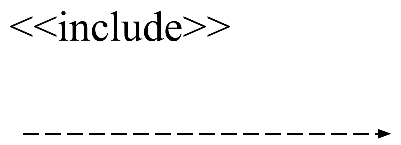


- <u>Generalization</u>: relationship between one general use case and a special use case (used for defining special alternatives) Represented by a line with a triangular arrow head toward the parent use case.

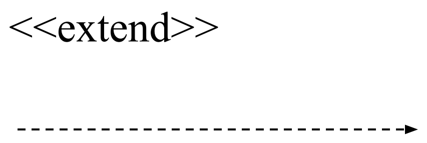


Use-Case Diagrams

Include: a dotted line labeled <<include>> beginning at base use case and ending with an arrows pointing to the include use case. The include relationship occurs when a chunk of behavior is similar across more than one use case. Use “include” in stead of copying the description of that behavior.



Extend: a dotted line labeled <<extend>> with an arrow toward the base case. The extending use case may add behavior to the base use case. The base class declares “extension points”.



Use-Case Diagrams

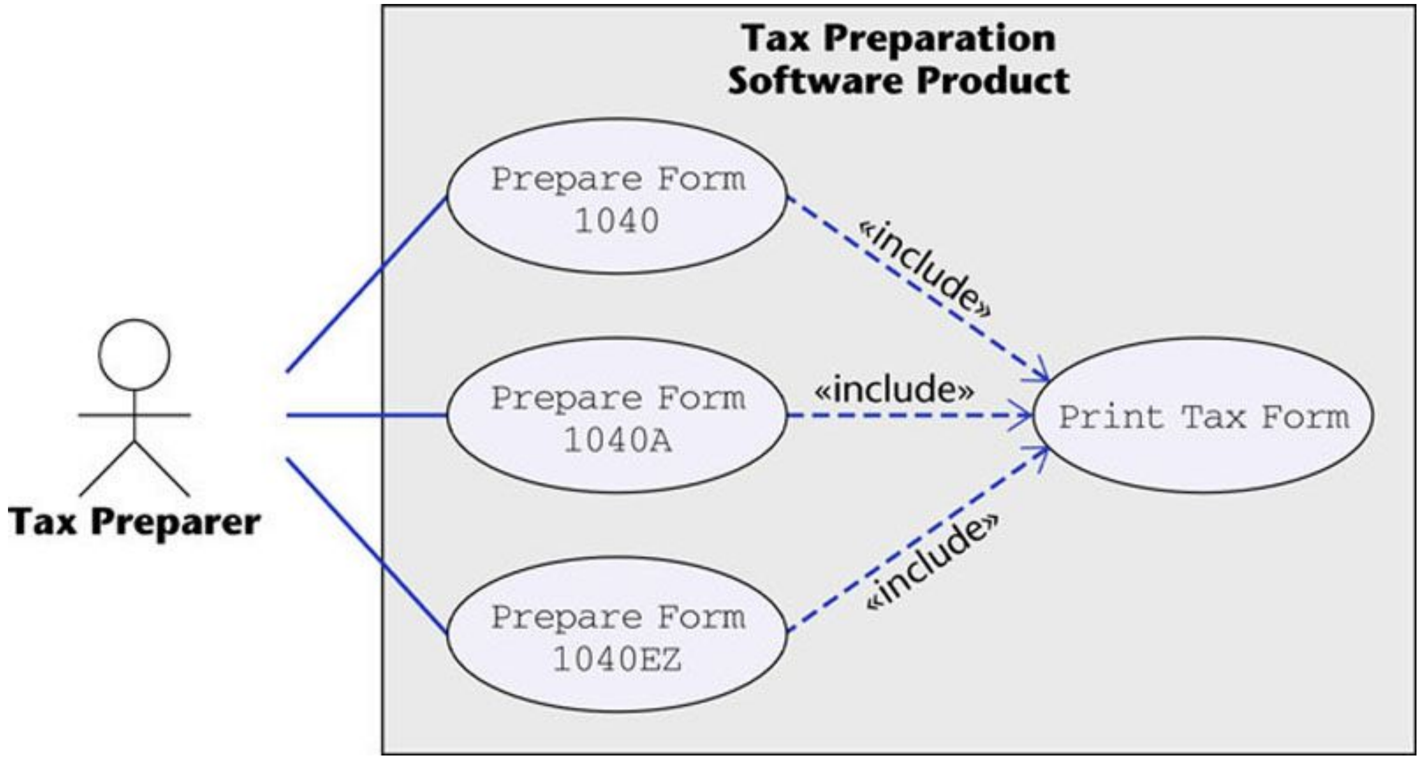
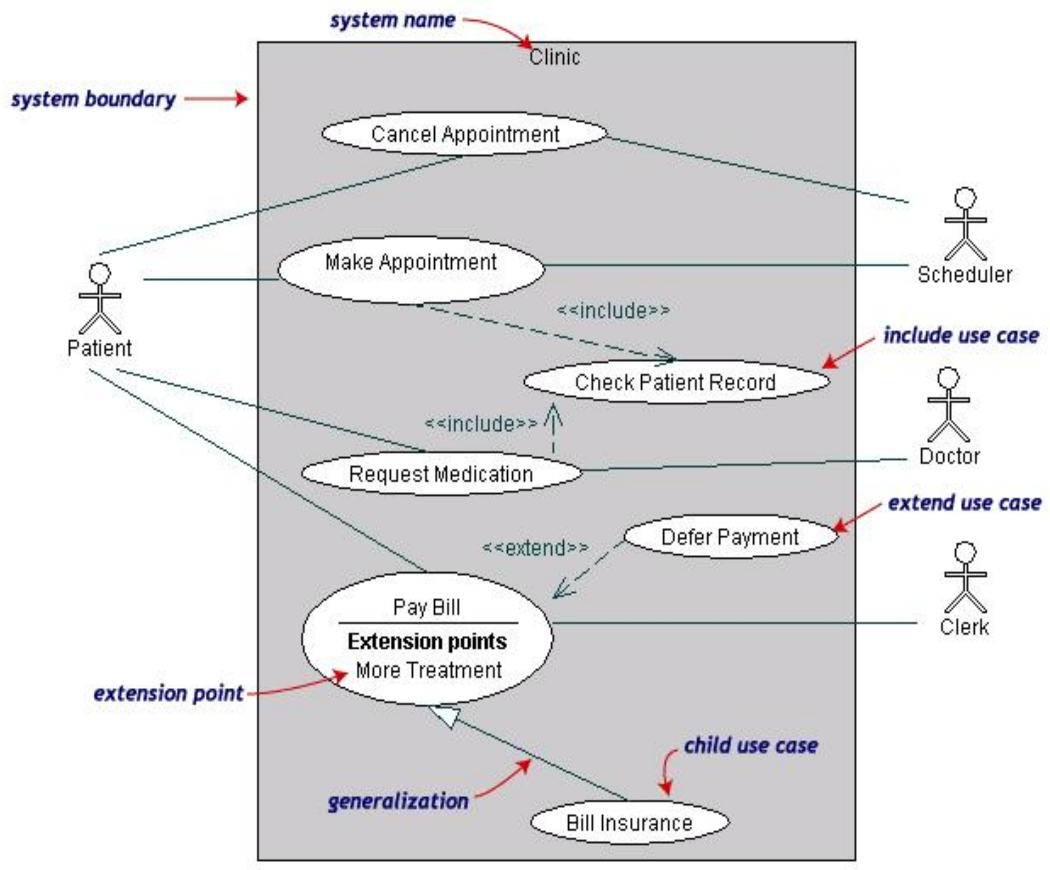


Figure 16.12

The McGraw-Hill Companies, 2005

Use-Case Diagrams

- Both **Make Appointment** and **Request Medication** include **Check Patient Record** as a subtask (include)
- The **extension point** is written inside the base case **Pay bill**; the extending class **Defer payment** adds the behavior of this extension point. (extend)
- **Pay Bill** is a parent use case and **Bill Insurance** is the child use case. (generalization)



(TogetherSoft, Inc)