Please start Visual Paradigm

Next class: Read UML chapters 10-12, 14, 15. Exercises due after class
Unified Modeling Language

- Graphical, object-oriented modeling language. Its uses are:
  - Sketch language to define system requirements
  - Blueprint language for system design
  - Implementation language to automatically generate software
- Open standard, managed by Object Management Group
  - Many implementations of UML (Microsoft, IBM, Visual Paradigm)
- Why is UML in wide use?
  - Speeds up requirements and design processes
  - Lessens information loss between requirements and design processes, and between design and implementation
  - Clearer than natural language
    - Provides a level of precision, but avoids details
    - Helps bridge language barriers in global projects
  - Supports iterative development (i.e., spiral model)
    - Supports both high level requirements/design in early spirals and detailed requirements/design later
  - Step toward analysts producing software without programmers
Unified Modeling Language uses

• Requirements:
  – **Use case diagrams**, which show multiple use cases or scenarios used to define system requirements
    • A use case is a sequence of operations performed by a system or person that produces a measurable result for an actor
    • Use cases are initiated by a user wanting to do something
    • Use cases record all possible events in system to achieve actor goals
  – **Component diagrams**, which show the hardware and software components of the system (what kind, how many, where...)
  – **Class diagrams**, which show multiple objects or things in a system, and the relationships between them
    • Derived from data models, which we cover in next unit

• Design:
  – More detailed use case, class, and component diagrams
  – **Activity and/or sequence diagrams**, used to model workflows, to find related or duplicate processes that can be generalized
  – **State diagrams** for complex objects

• Implementation:
  – Class, state and other diagrams (vendor-specific)
Use cases

• Use case modeling is process of describing behavior of system from external point of view
  – Use case describes what a system does, not how it does it
  – Emphasizes modeling external, not internal, point of view to focus on requirements, not implementation
  – Captures requirements of system as list of structured scenarios
  – Use cases are the basic unit of requirements definition
  – Actor in use case can be person, computer/device or external system
    • Actor represents group of users or role, not specific individual
Use case example 1

Nothing is implied by the order of the use cases; they are not sequential
Use case example 2

**Use case/scenario**

- Present valid ticket
- Present invalid ticket
- Has no ticket or money
- Purchase ticket onboard
- Single ride
- Monthly pass
- Enforcement
- Conduct
- Passenger

**Included/related scenario**

<<Include>>

**Association**

Actor/role (can be system)

---

*“Include” vs “extend” discussed later*
Use case exercise

• Exercise: medical appointment management system
  – List the use cases (scenarios) in a medical visit
    • May see nurse, doctor and/or lab technician
    • May have tests (blood work, urine sample, etc.)
    • May have immunizations/shots
    • May get prescription
    • May get referral to specialist
    • Etc.
  – Draw use cases, actors, “include” or generalize to link related use cases
    • Don’t use ‘extends’ for now, to keep it simple
Use case solution

- Prescribe shots
- Prescribe medication
- Referral
- Evaluate tests
- Update patient records
- Review patient records

- Perform test
- Prescribe tests
- Give shots
- Prescribe shots
- Prescribe medication
- Blood
- Urine
- Other

- Lab tech
- Nurse
- Doctor

Image by MIT OpenCourseWare.
# Summary - use case diagrams

<table>
<thead>
<tr>
<th>Element</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use case</td>
<td>Set of operations performed by/in system that produces measurable result for an actor</td>
</tr>
<tr>
<td>Actor</td>
<td>Set of roles that users (can be a system) play</td>
</tr>
<tr>
<td>System</td>
<td>Boundary between a software/hardware/manual system and other actors or systems</td>
</tr>
<tr>
<td>Association</td>
<td>Participation of an actor in a use case</td>
</tr>
<tr>
<td>Generalization</td>
<td>Relationship between general and more specific actor or use case. Arrow points to general use case or actor</td>
</tr>
<tr>
<td>Include</td>
<td>Variation on base use case</td>
</tr>
<tr>
<td>Extend</td>
<td>Some modelers (not us) make the following distinction: Include is used when a common use case is inserted in two or more base cases: e.g., “login” used both by “make reservation” and “cancel reservation” Extend is used when a variation is inserted in only one base case: e.g., “make multiple reservations” extends “make reservation”</td>
</tr>
</tbody>
</table>
Component diagram exercise

- Component diagrams
  - List all the “things” in a system
  - Used to set system scope, prerequisites, stakeholders
- Medical appointment components:
  - Labs, lab equipment
  - Offices for nurse/doctor, and their equipment
  - Computer systems
  - Etc.
- Draw component diagram for medical appointment
- Use only components and generic connectors in the UML diagram
  - Focus on the lab equipment, databases, and other systems
  - Begin to make decisions on what is included within your system scope and what is excluded
Component diagram example

- Component: Lab equipment
- Component: Lab control and data system
- Component: Medical records system

Association: Lab equipment and Lab control and data system
Association: Lab control and data system and Medical records system
Component diagram solution

- Lab equipment
- Lab control and data system
- Medical records system
- Paper records
- Billing system
- Appointment/time management system
Summary

• **Use cases**
  – Use case is a scenario or set of steps to achieve a goal
  – Use case diagram contains **all** relevant scenarios for a system (or system component)
  – Diagram helps capture full list of scenarios, and summarizes them compactly
  – **Write short (1 page)** descriptions of each scenario next, after creating use case diagrams
    • Or build use cases from written requirements if present
  – Use UML interactively with stakeholders in setting requirements
  – A text document listing scenarios wouldn’t work

• **Component diagrams**
  – List all the “things” in a system
  – Used to set system scope, prerequisites, stakeholders
  – Usually much simpler to create than use case diagram