Object Oriented Analysis and Design

UML

Unified Modeling Language

UML Notation

- Grady Booch
- James Rumbaugh
- Ivar Jacobson
- Rational Software Corporation

UML - is OMG's (Object Management Group) most-used specification

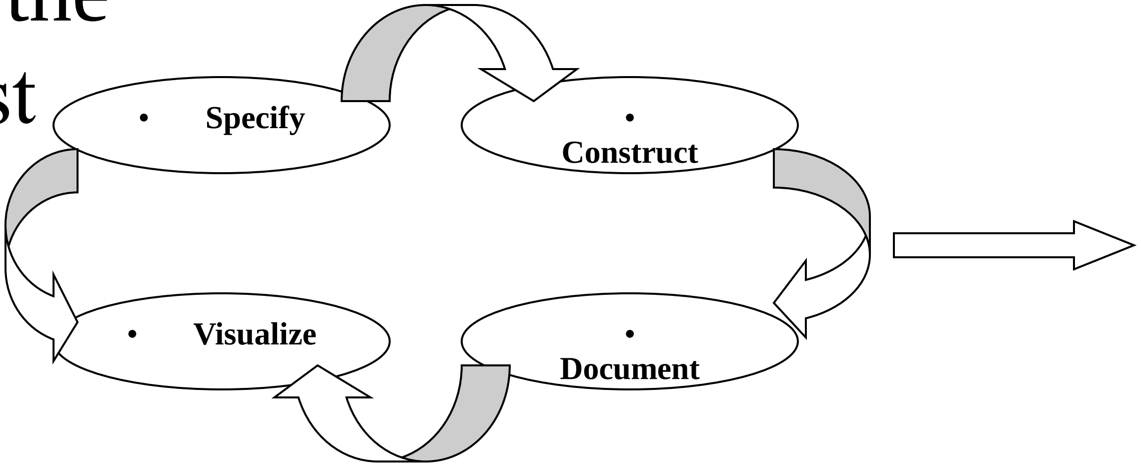
What is UML?

- Is a language for constructing and documenting software system artifacts

- Is the standard language for visualizing, specifying, constructing and documenting software artifacts through graphical tools.

UML Unification

Result of unifying the information systems and technology and the industry's best engineering practices



•Programming Language

•Tools Specification

•Process

Features in UML Tools

- **UML Diagram Support**
  - should support all the nine diagrams
- **Forward Engineering**
  - must translate diagram into actual source code (classes) with the methods stubbed out
- **Reverse Engineering**
  - loads all the files of the application/system, identifies dependencies between the various classes, and essentially reconstructs the entire application structure along with all the relationships between the classes

Features in UML Tools

- **Round-trip Engineering**
  - synchronize the model with the changes in the application code
- **Documentation**
- **Version Control**
  - maintaining versions and baselines of the system design
- **Collaborative Modeling Environment**
  - compare different versions designs for differences
  - merge different versions of a design

Features in UML Tools

- **Integration with popular Integrated Development Environments (IDE)**
- **Test script generation**
- **Model View Controller (MVC) modeling**

Types of UML Diagrams

- class (package)
- object
- use case
- sequence
- collaboration
- statechart
- activity
- component
- deployment

Use Case Diagram

- The Use case diagram is used to identify the primary elements and processes that form the system.
- The primary elements are termed as actors and the processes are called use cases
- The Use case diagram shows which actors interact with each use case

Who creates use case diagram?

Domain experts and business analysts should be involved in writing use cases for a given system

When use case diagram is created?

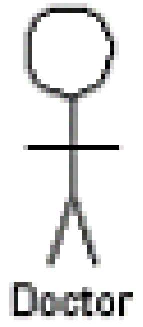
When the requirements of a system need to be captured

Actor

- An actor portrays any entity (or entities) that performs certain roles in a given system

- For example, for modeling a banking application, a customer entity represents an actor in the application. Similarly, the person who provides service at the counter is also an actor

- "patients visit the doctor in the clinic for medical tests," "doctor" and "patients" are the actors



Actor

- An actor is something or someone who supplies a stimulus to the system.
- An actor cannot be controlled by the system and is defined as being outside the system.
- An actor is often thought of as a role, rather than an actual person. A single person in the real world can be represented by several actors if they have several different roles and goals in regard to a system.

Actor

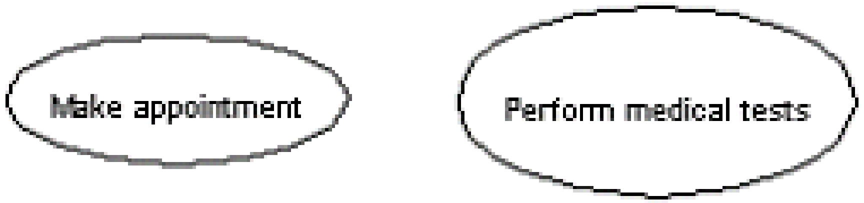
- Primary Actors interact directly with a system to achieve their goals.
- Supporting actors may be humans or systems called in to support the Primary Actor.
- Stakeholders can also be modeled as actors. They do not directly interact with the system but they are affected by the success of Primary Actor interactions.
- *Active* actors initiate interactions with a system, while *passive* actors act as targets of requests or are activated by the system.

Project Stakeholders

- direct user
- indirect user
- manager of users
- senior manager
- operations staff member
- the "gold owner" who funds the project
- support (help desk) staff member
- auditors
- your program/portfolio manager
- developers working on other systems that integrate or interact with the one under development
- maintenance professionals potentially affected by the development and/or deployment of a software project

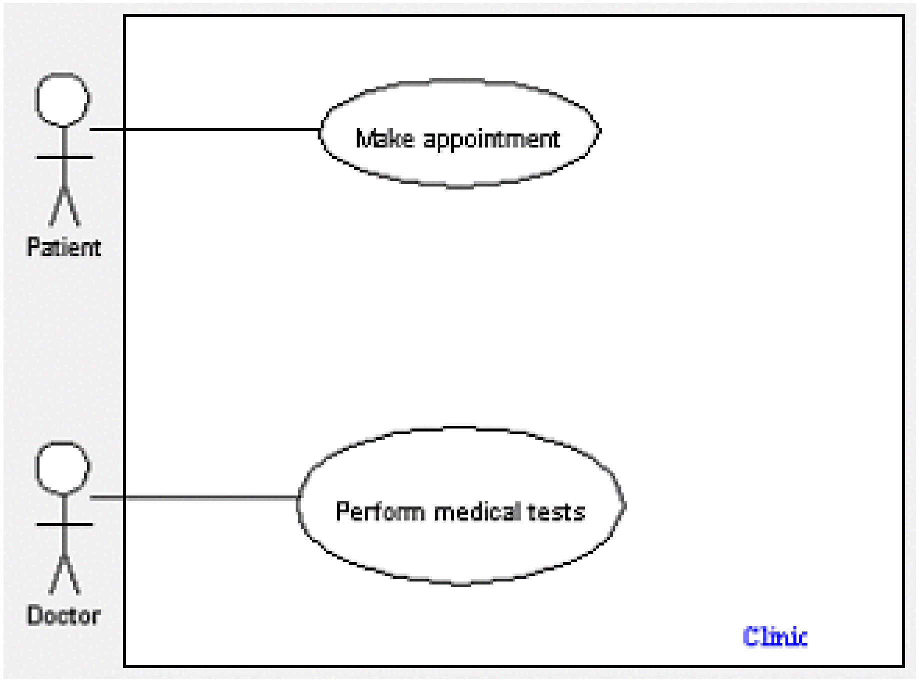
Use Case

- A use case is a visual representation of a distinct business functionality in a system
- Need to ensure that the business process is discrete in nature
- For example, two uses cases: "Make appointment" and "Perform medical tests" in the use case diagram of a clinic system
- A business process such as "manage patient records" can in turn have sub-processes like "manage patient's personal information" and "manage patient's medical information."



System Boundary

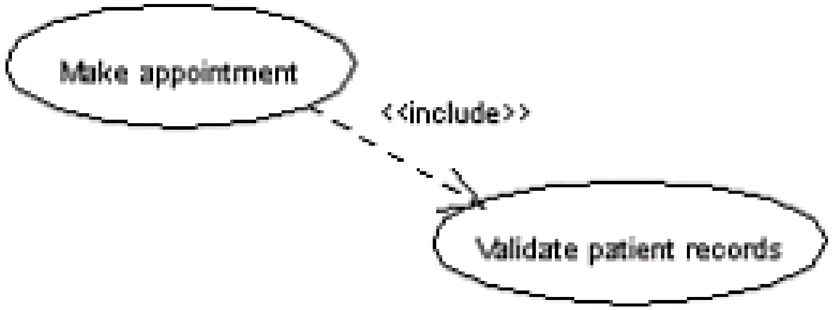
- A system boundary defines the scope of what a system will be
- For large and complex systems, each of the modules may be the system boundary



Relationships in use cases (dependency)

Include

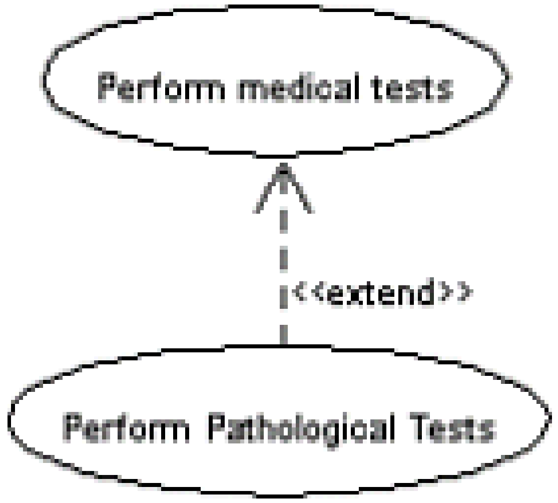
- When a use case is depicted as using the functionality of another use case in a diagram
- directed arrow having a dotted shaft
- The tip of the arrowhead points to the child use case and the parent use case is connected at the base of the arrow
- Uses the stereotype <<include>>



Relationships in use cases (extension)

Extend

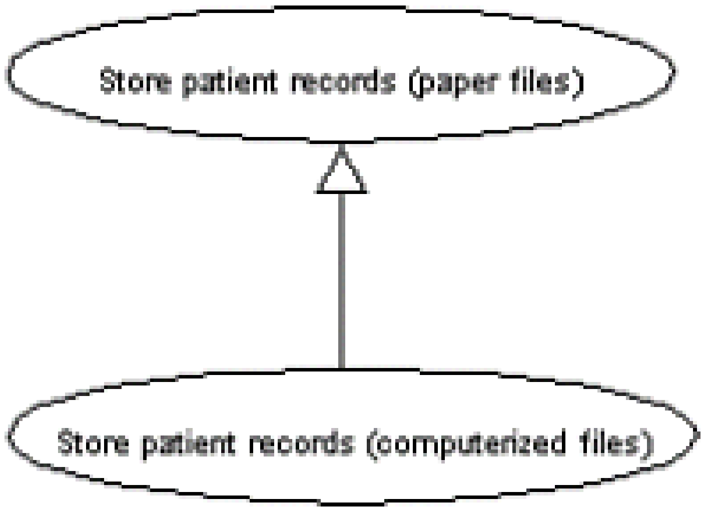
- The child use case adds to the existing functionality and characteristics of the parent use case
- directed arrow having a dotted shaft
- The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow
- Uses the stereotype <<extend>>



Relationships in use cases (specialization)

Generalization

- The child use case in the generalization relationship has the underlying business process meaning, but is an enhancement of the parent use case
- directed arrow with a triangle arrowhead
- The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case


Store patient records (paper files)
Store patient records (computerized files)

Subtle difference between a generalization relationship and an extend relationship

Generalization relationship - Parent use case can be replaced by the child use case without breaking the business flow

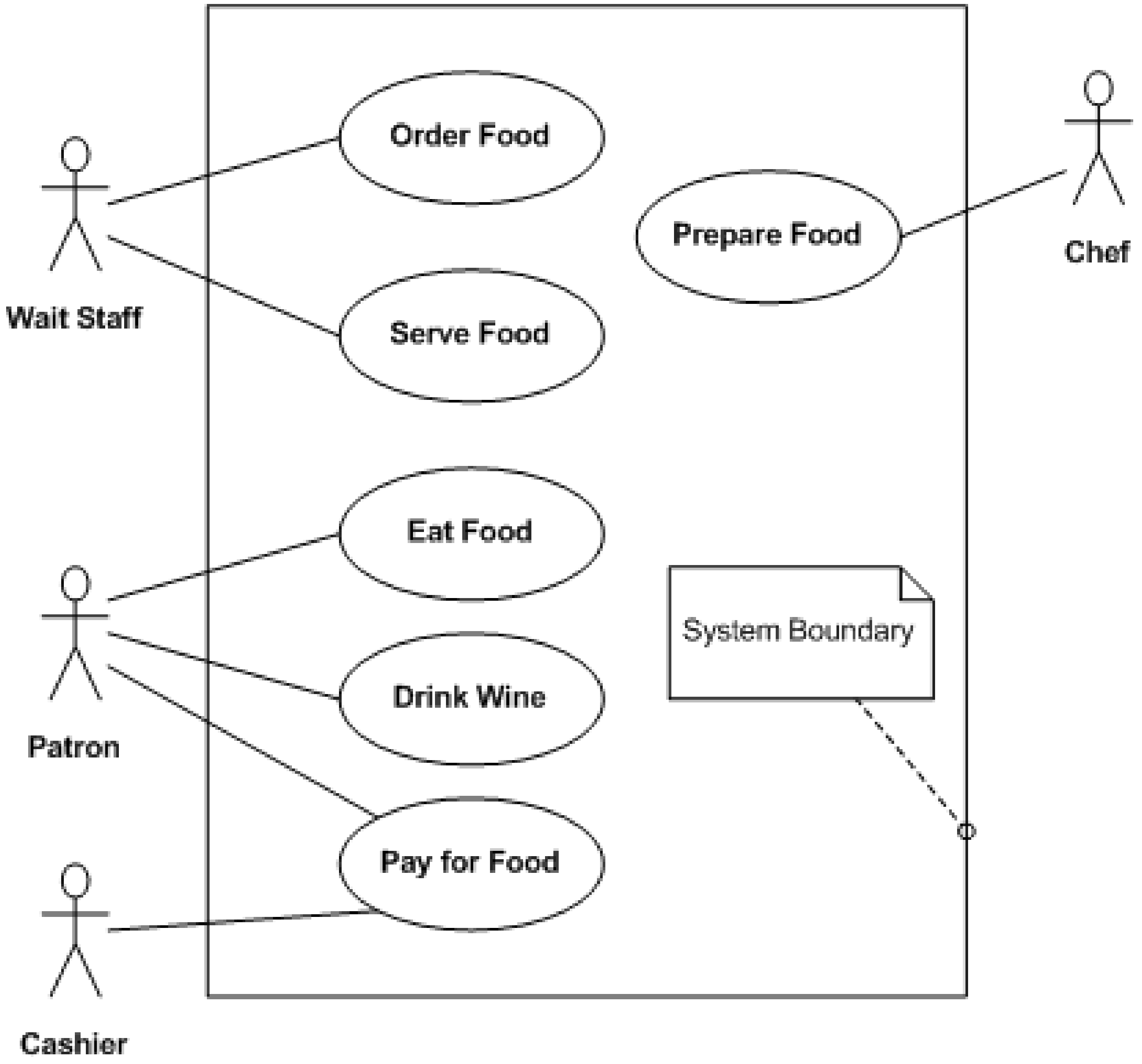
Extend relationship - Child use case enhances the functionality of the parent use case into a specialized functionality.

Parent use case cannot be replaced by the child use case

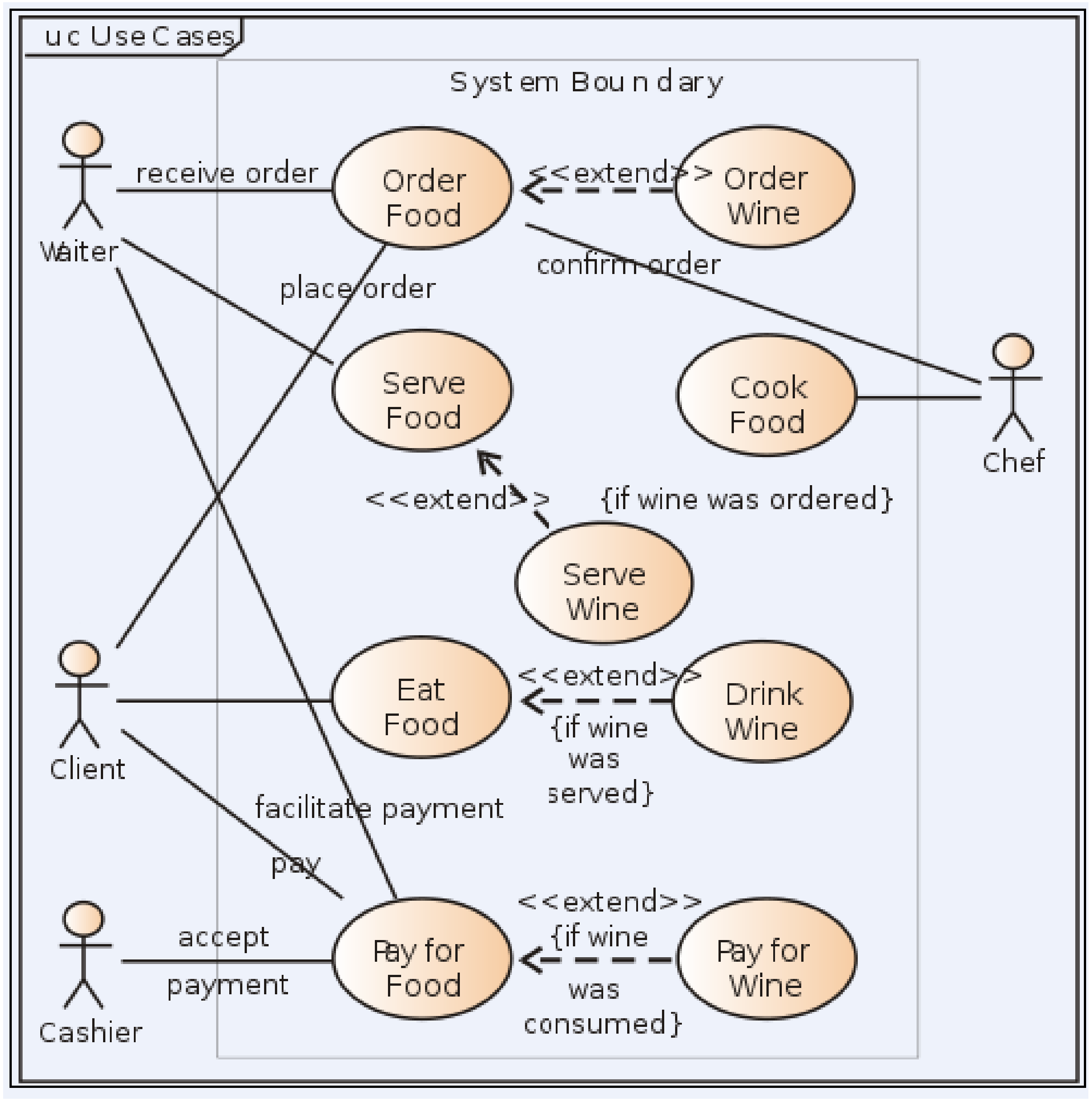
Difference between <<include>> and <<extend>>

- To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write ***include*** followed by the name of use case you want to include.
- Modelers use the «**extend**» relationship to indicate use cases that are "optional" to the base use case. Depending on the modeler's approach "optional" may mean "potentially not executed with the base use case" or it may mean "not required to achieve the base use case goal".

UML Use Cases for a Simplistic Restaurant Model



UML Use Cases for a Simplistic Restaurant Model



Case study-Courseware Management System

Courses and Topics that make up a course

Tutors who teach courses

Course administrators who manage the assignment of the courses to tutors

Calendar or Course Schedule is generated as a result of the assignment

Students who refer to the Course schedule or Calendar to decide which courses they wish to take up for study