UML use cases

Some content adapted Applying UML and patterns by Craig Larman

What is use case modeling?

- Use case model: a view of a system that emphasizes the behavior as it appears to outside users. A use case model partitions system functionality into transactions (‘use cases’) that are meaningful to users (‘actors’).

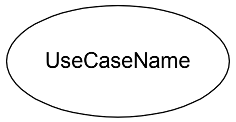
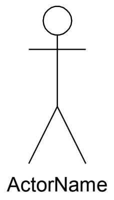
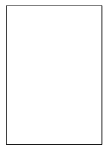
Use Cases

- *Use cases* tell stories of actors using a system.
- A *use case scenario* is a sequence of actions a system performs that yields an observable result of value to a particular actor.
- It describes an end-to-end process --- from when a user starts to use the system for a particular purpose until they are done (for that purpose).
- *Use cases* are an excellent mechanism to express functional requirements. They emphasize thinking from the viewpoint of the users.
- We use *use cases* to "drive" the process. Our goal during development is to make specific use cases operational.

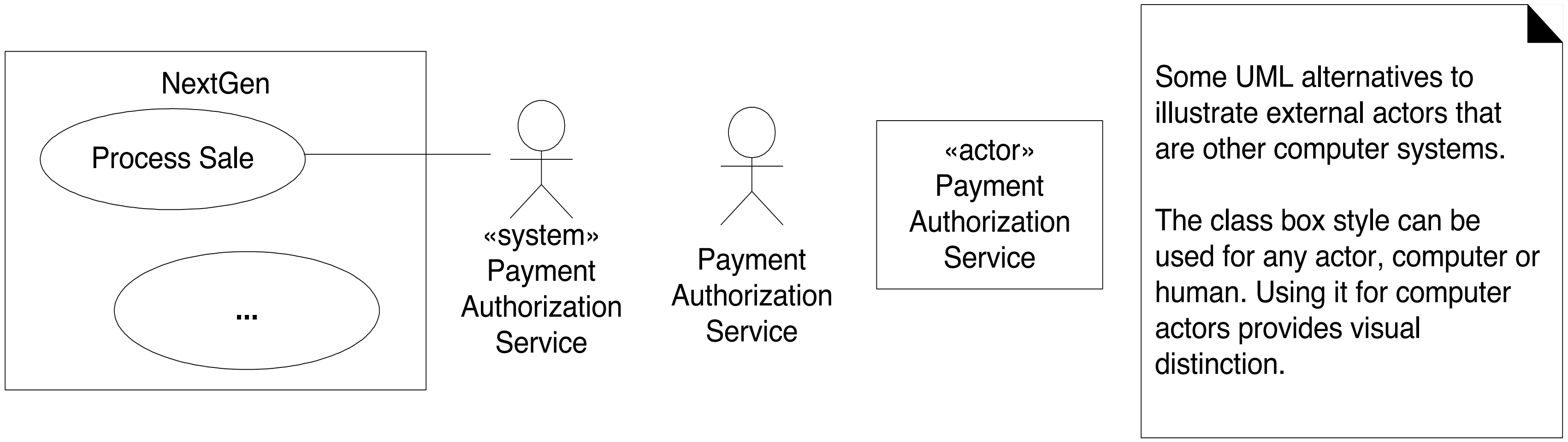
Use Cases in UML 2.X

- Use cases are associated with subjects
  - A subject can be a system, a subsystem in a system, or a class
- A use case describes interactions between users (clients) and a subject

Use Case Modeling: Core Elements

Construct	Description	Syntax
use case	A sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system.	
actor	A role played by an entity that interacts with the subject (e.g., system, subsystem, class).	
System/subject boundary	Represents the boundary between the subject and the actors who interact with the subject.	

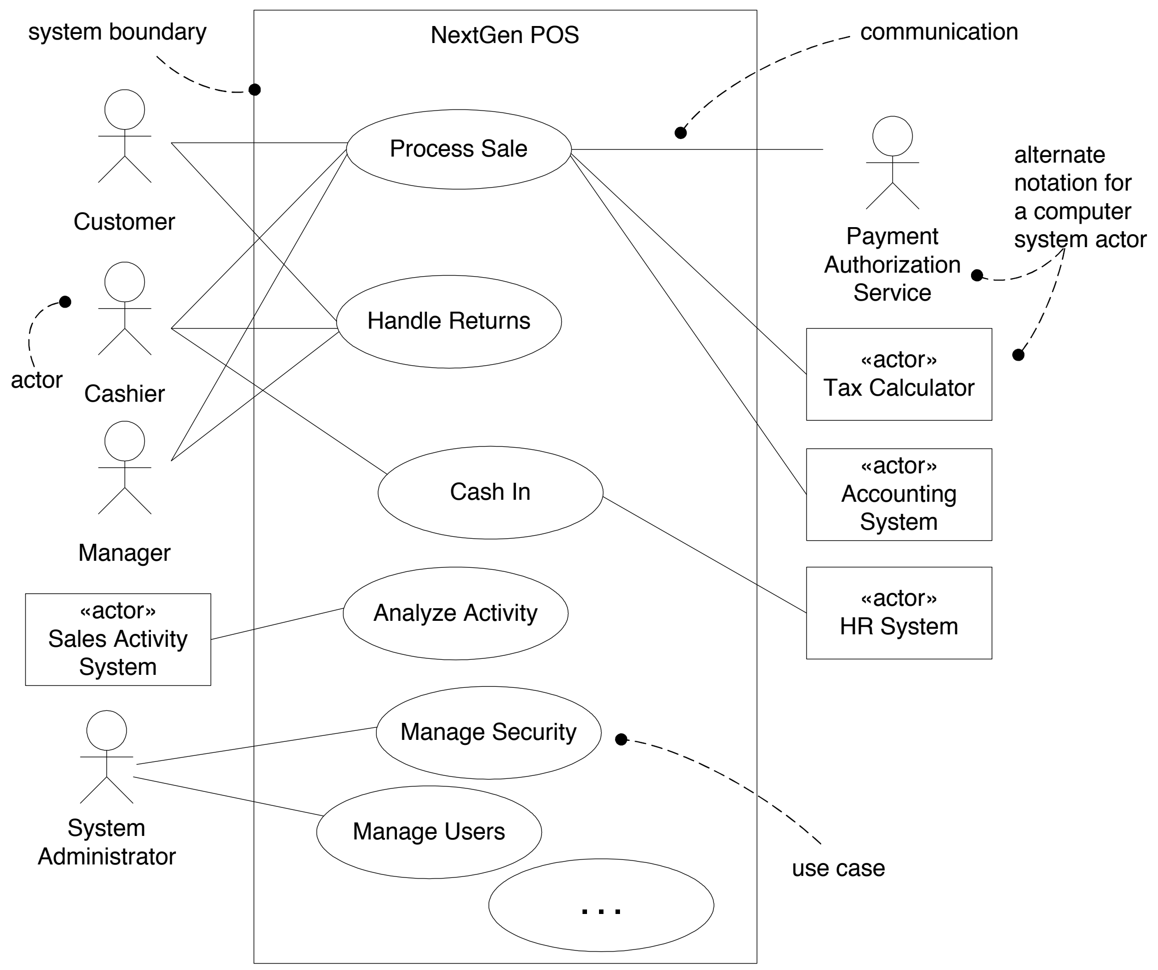
Depicting actors



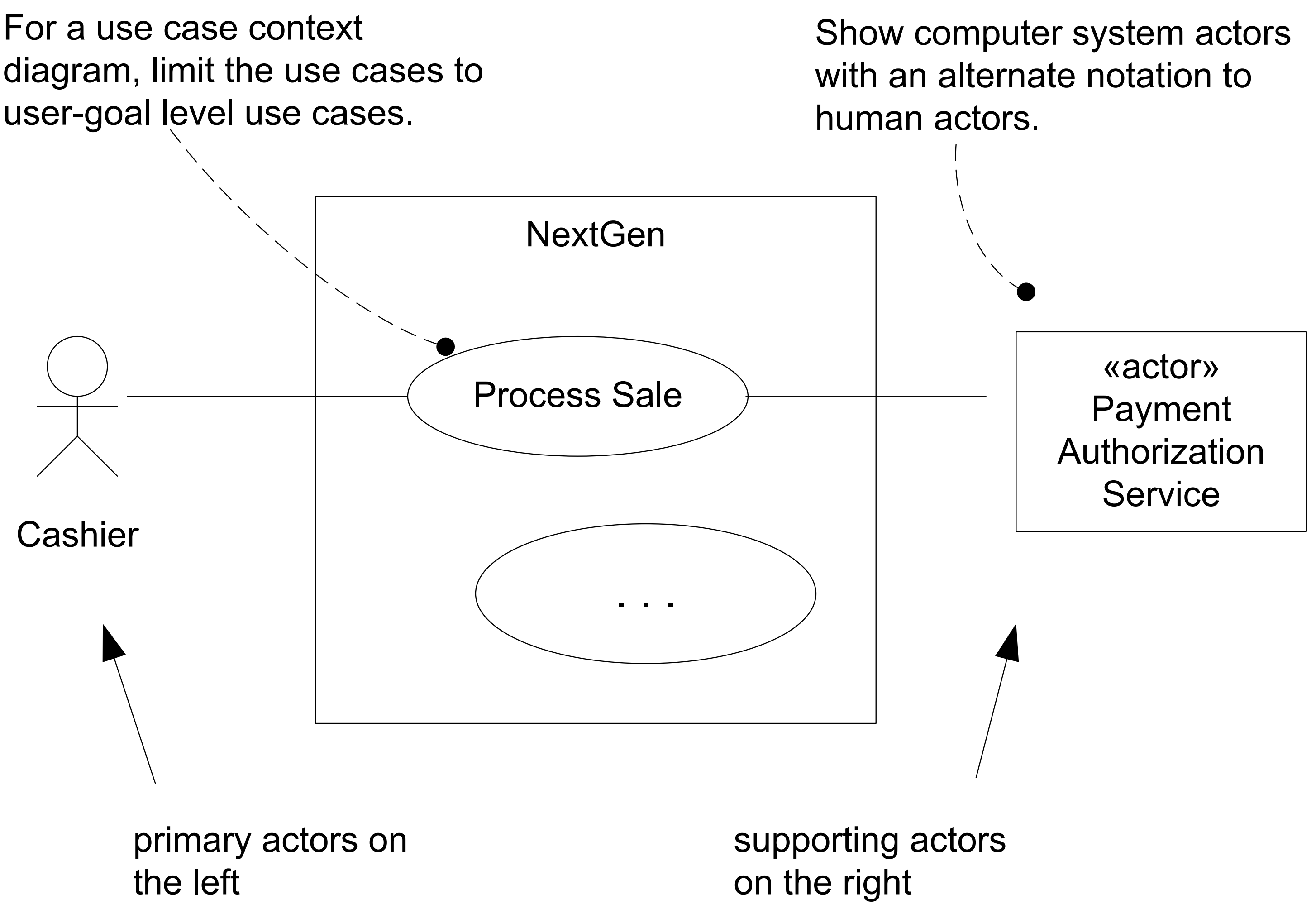
Some UML alternatives to illustrate external actors that are other computer systems.

The class box style can be used for any actor, computer or human. Using it for computer actors provides visual distinction.

Use Case Diagram



Use Case Diagram Style


For a use case context diagram, limit the use cases to user-goal level use cases.
Show computer system actors with an alternate notation to human actors.
NextGen
Process Sale
. . .
Cashier
«actor» Payment Authorization Service
primary actors on the left
supporting actors on the right

Use Cases as Requirements

- Use cases can be used to capture functional requirements
  - System attributes associated with a system operation can be documented in a use case
- Not all requirements can be captured by use cases
  - System attributes that span use cases are documented as supplementary requirements

Use Case Instance (Scenario)

- A **scenario** is a particular sequence of actions in a use case.
  - A use case is a related set of scenarios that yields an observable result of value to a particular actor
- A **use case instance** is an execution of a scenario.
  - Often use case instance and scenario are used synonymously in informal discussions.

Levels of Rigor

- **Brief**: One paragraph summaries of functionality
- **Casual**: Multiple paragraphs that cover multiple scenarios
- **Fully-dressed** (Detailed): Structured, detailed description of scenarios

Essential vs. Concrete Use Cases

- **Essential Use Cases** describe functionality in implementation independent terms
  - Requirements level use cases must be essential
- **Concrete Use Cases** describe external functionality in system dependent terms
  - Use cases can be used during design to document externally observable behavior of subsystems

Requirements Use Case Template - 1

Use Case Number	EU-xxxx : Indicates an essential use case, i.e., a use case that describes activity in system independent terms	
Use Case Name	*Enter name of Use Case.*	
Overview	*Describe the purpose of the Use Case and give a brief description.*	
Type	*Enter Use Case priority* (primary, secondary, optional)	
Actors	•*List all actors that participate in this Use Case. Indicate the actor that initiates the use case by placing "initiator" in brackets after the actor name. Also, indicate primary actors by placing "primary" in brackets after actor name.*	
Properties	***Performance:***	
	Security:	
	Other:	

Use Case Template – cont’ d

Pre-condition	*Enter the condition that must be true when the main flow is initiated. This should reference the conceptual model.*	
Flow	***Main Flow:*** *Steps should be numbered.*	
		Subflows: *Break down of main flow steps*
		Alternate Flows: *Include the post condition for each alternate flow if different from the main flow.*
Post Condition	*Enter the condition that must be true when the main flow is completed. This should reference the conceptual model. Include the following information in this section:*	
Cross References	*References to other Use Cases or textual requirements that relate to this Use Case.*	

Use case example

Use Case Number:	EU-0001
Use Case Name:	Withdraw Money
Overview:	Customer withdraws money from an ATM .
Type:	primary
Actors:	Customer
Pre-condition:	Customer has selected withdraw option
Main flow:	1. System requests customer PIN 2. Customer keys in PIN and submits to system 3. If the PIN is valid ten the system acknowledges valid entry and asks for amount to be withdrawn 4. Customer enters amount to be withdrawn 5. If amount entered in less than amount in Customer's account, system dispenses the cash, else system informs customer that amount cannot be withdrawn
Alternate flow	3. If the PIN is invalid then the use case is restarted. If this occurs 3 consecutive times then the system cancels the transaction and prevents the customer from interacting for 60 seconds
Alternate flow	2. The customer can cancel the PIN validation at anytime. Cancellation causes the use case to restart
Alternate flow	2. The customer can clear and reenter the PIN any number of times before submitting the PIN
Post-condition	True
Cross-reference:	Validate PIN

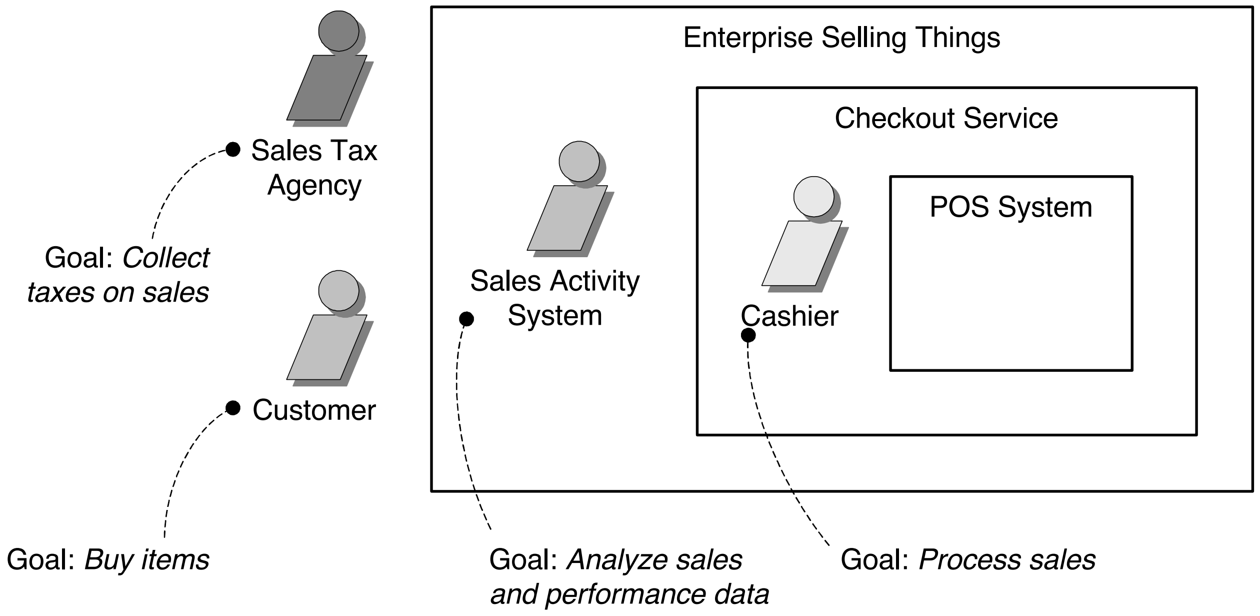
Actor Types

- **Primary**: actor whose goal is accomplished by the use case
- **Supporting**: actor that provides services to the system
  - E.g., authorization service
- **Offstage**: an actor that has an interest in the use case but is not primary or supporting
  - E.g., regulating agency

Use Case Scope

A use case should describe end-to-end functionality

- Should describe a task carried in response to an event generated by an actor that produces a result of value to a subset of its actors and leaves the system in a stable state (one in which it is not waiting for a restricted set of inputs)



Developing Use Cases

- Scope system and identify primary actors that interact with the system
- Determine goals of primary actors (can be documented in an Actor-Goal list)
- For each actor, consider the ways that the actor typically interacts with the system to accomplish goals
- Consider exceptional behaviors

Use Case Modeling Tips

- Writing essential use cases: Focus on intent
  - Keep user interface terms out
  - Ask "what is the goal?"
- Write "black-box" use cases
  - Do not describe internal operations (e.g., storing to a data base)
- Focus only on interactions between system and actors
  - Ignore interactions between actors
- Focus on text description
  - Use diagrams for presentation purposes only
- A use case diagram should
  - contain only use cases at the same level of abstraction
  - include only required actors

How do I know I have a good use case?

- Use case should describe an activity that yields an observable result of value to an actor.
- A use case can describe an elementary business process: a sequence of tasks performed to handle a business event
- Use cases are typically not single steps or single low-level actions.

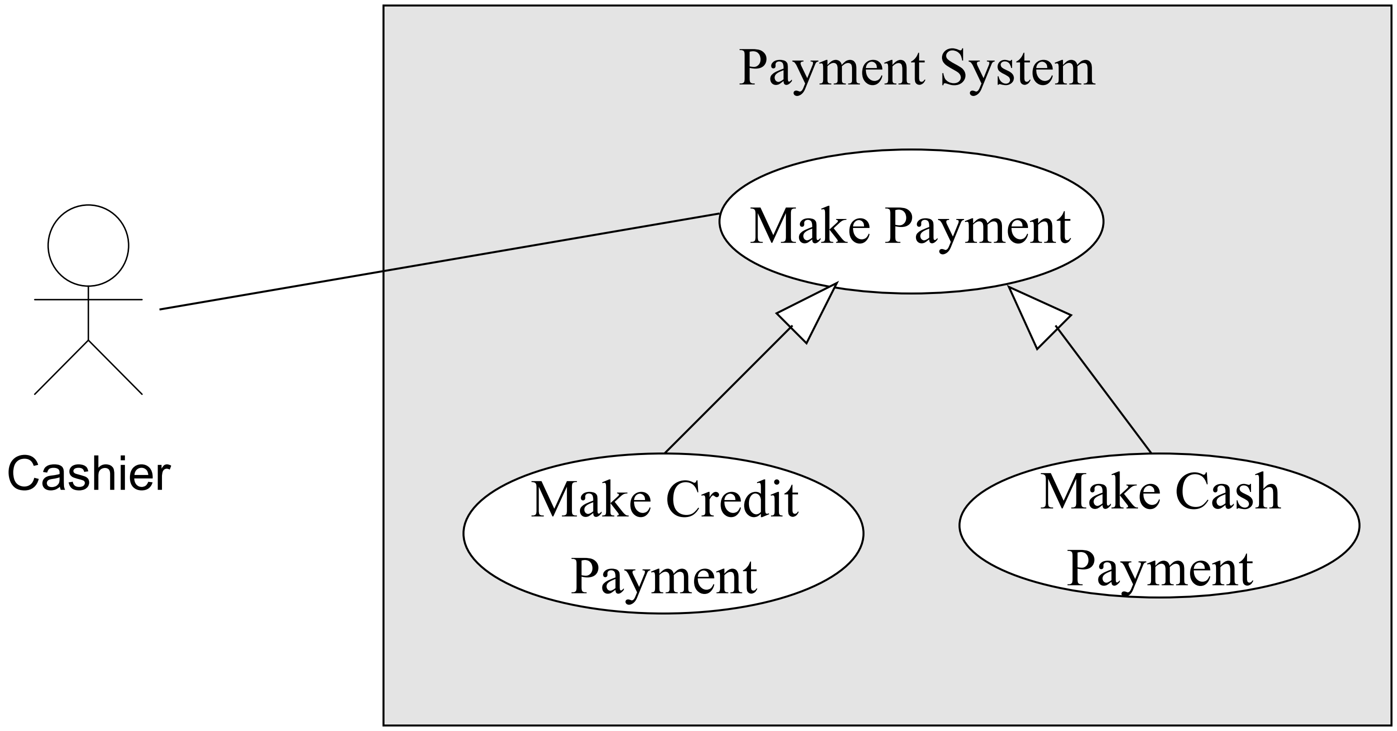
Organizing Use Cases

- Specializing/generalizing use cases
- Including use cases
- Extending use cases

Specializing Use Cases

- Generalizing/Specializing use cases
  - A specialized use case inherits the behavior (sequence of actions) of its parent(s)
  - A specialized use case can override some of the behavior of its parent(s). It can also add to the behavior
  - A specialized use case can be used anywhere the general use case is expected.

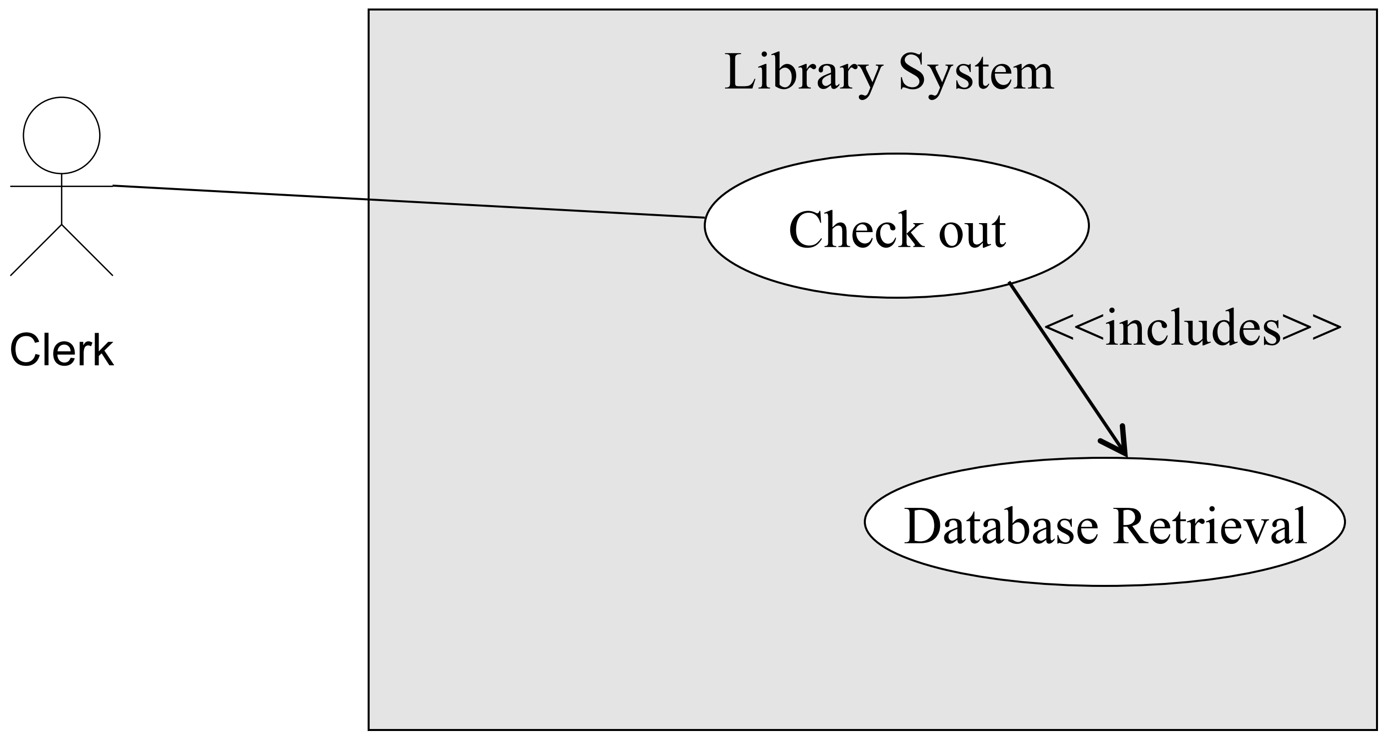
Specializing Use Cases



Including Use Cases

- A use case can include another use case at a specified location
  - Used to avoid writing the same flow of events across a number of use cases.
  - The included use case must not be a stand-alone use case.

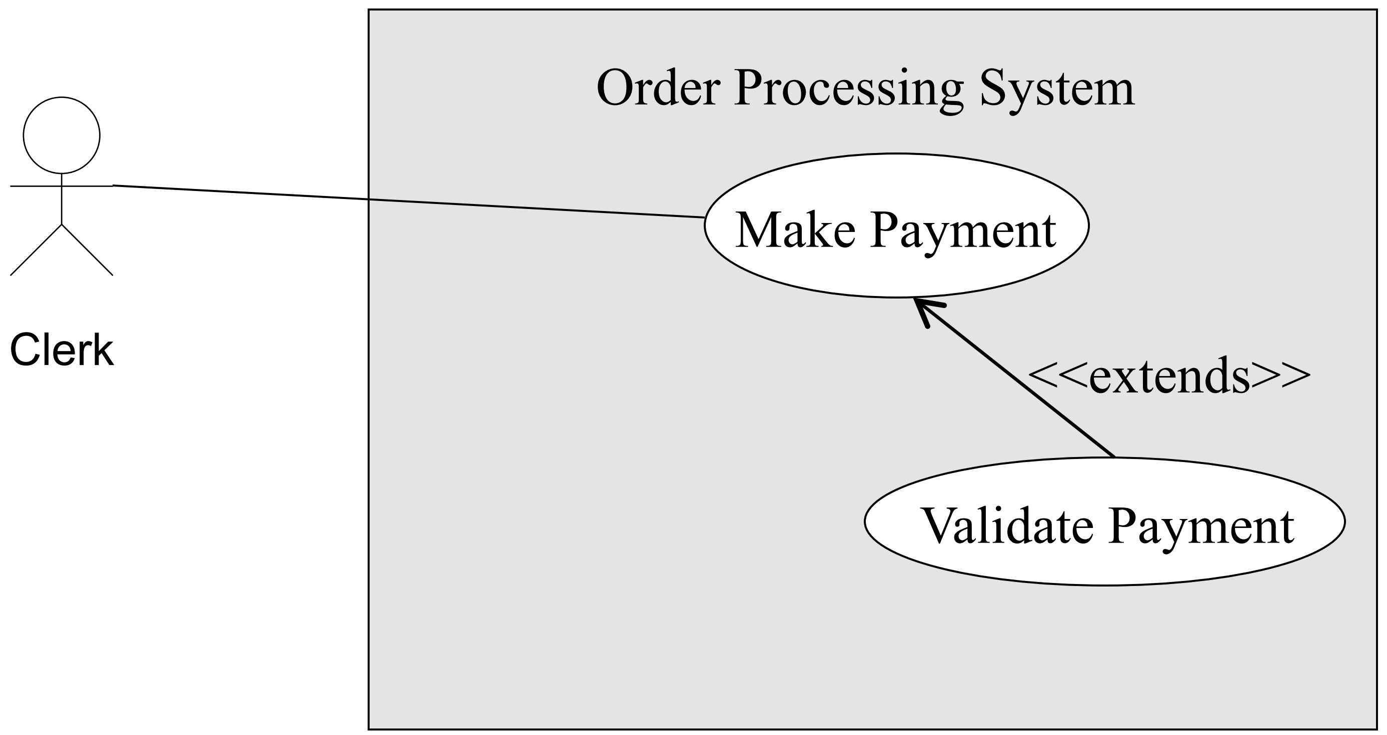
Including Use Cases



Extending Use Cases

- A use case can extend a base use case by incorporating additional behavior at specified locations of the base use case
  - The base use case can act as a stand-alone use case.
  - The base use case can only be extended at specified points called *extension points*.
  - Often used to separate optional behavior from mandatory behavior
  - Also used to model a separate flow that is executed under certain conditions

Including Use Cases



Summary

- Use case model the behavior of the system
  - Functional requirements are mapped to use cases
  - Non-functional requirements can be specified as constraints (not done in most of the cases)