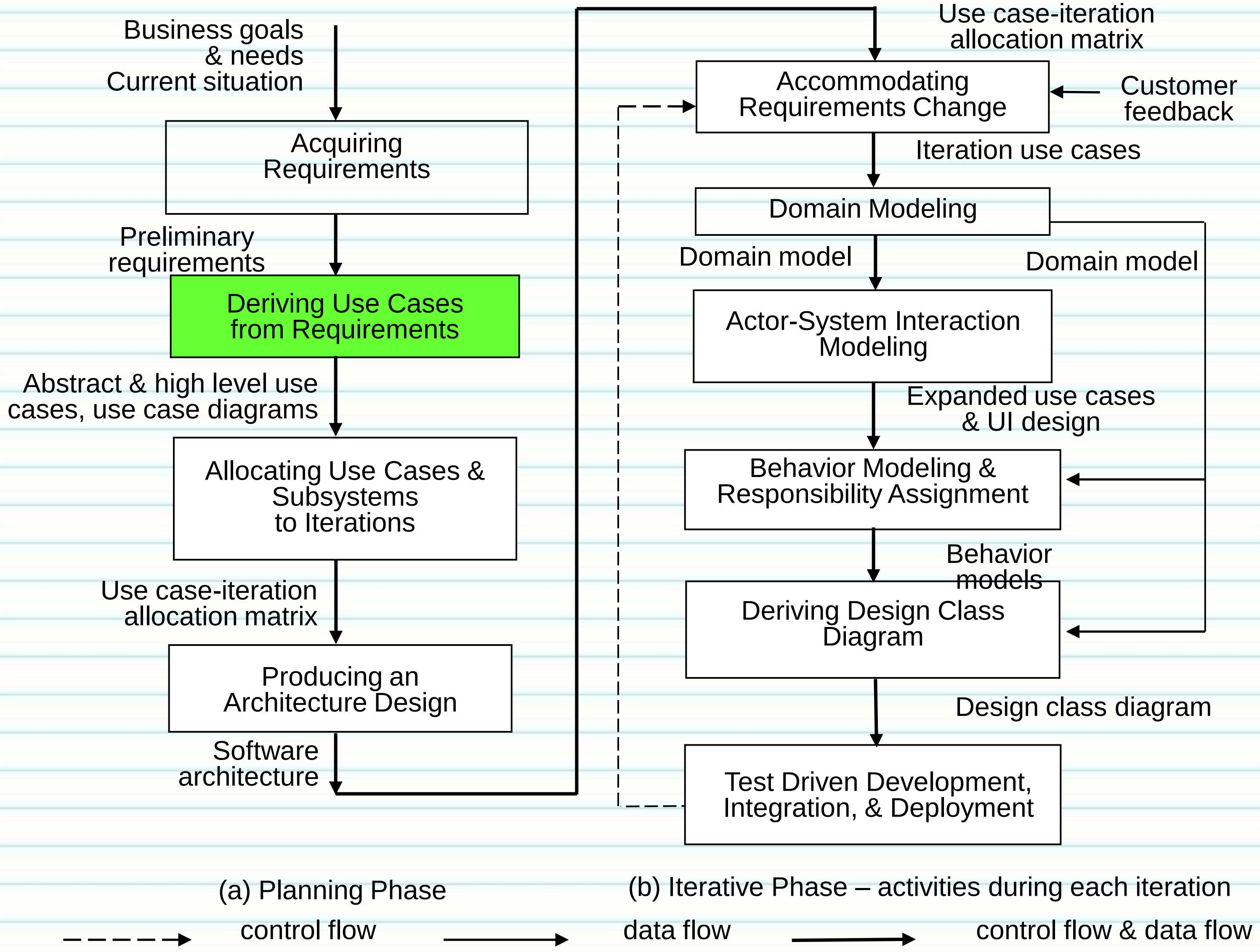
Chapter 7 – Deriving Use Cases from Requirements

Dr John H Robb, PMP
UT Arlington
Computer Science and Engineering

Key Takeaway Points

- A use case is a business process; it begins with an actor, ends with the actor, and accomplishes a business task for the actor.
- Use cases are derived from requirements and satisfy the requirements.
- Planning the development and deployment of use cases and subsystems to meet the customer’s business needs and priorities.

Deriving Use Cases in the Methodology Context


Business goals & needs Current situation
Acquiring Requirements
Preliminary requirements
Deriving Use Cases from Requirements
Abstract & high level use cases, use case diagrams
Allocating Use Cases & Subsystems to Iterations
Use case-iteration allocation matrix
Producing an Architecture Design
Software architecture
Use case-iteration allocation matrix
Accommodating Requirements Change
Customer feedback
Iteration use cases
Domain Modeling
Domain model
Domain model
Actor-System Interaction Modeling
Expanded use cases & UI design
Behavior Modeling & Responsibility Assignment
Behavior models
Deriving Design Class Diagram
Design class diagram
Test Driven Development, Integration, & Deployment
(a) Planning Phase
(b) Iterative Phase – activities during each iteration
control flow
data flow
control flow & data flow

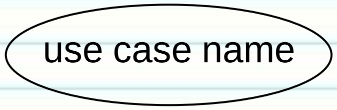
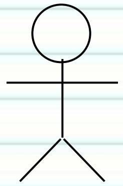
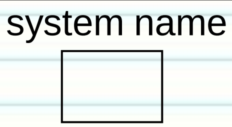
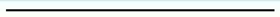
What Is a Use Case?

- ***A use case is a business process.***
- A use case must be initiated by an actor.
- A use case must end with the actor.
  - The actor explicitly or implicitly acknowledges the accomplishment of the business task.
- A use case must accomplish a business task (for the actor).

What Is an Actor?

- An actor denotes a *business role* played by (and on behalf of) a set of business entities or stakeholders.
- Actors are not part of the system.
- Actors interact with the system.
- Actors are often human beings but can also be a piece of hardware, a system, or another component of the system.
- Actors initiate use cases, which accomplish business tasks for the respective actors.

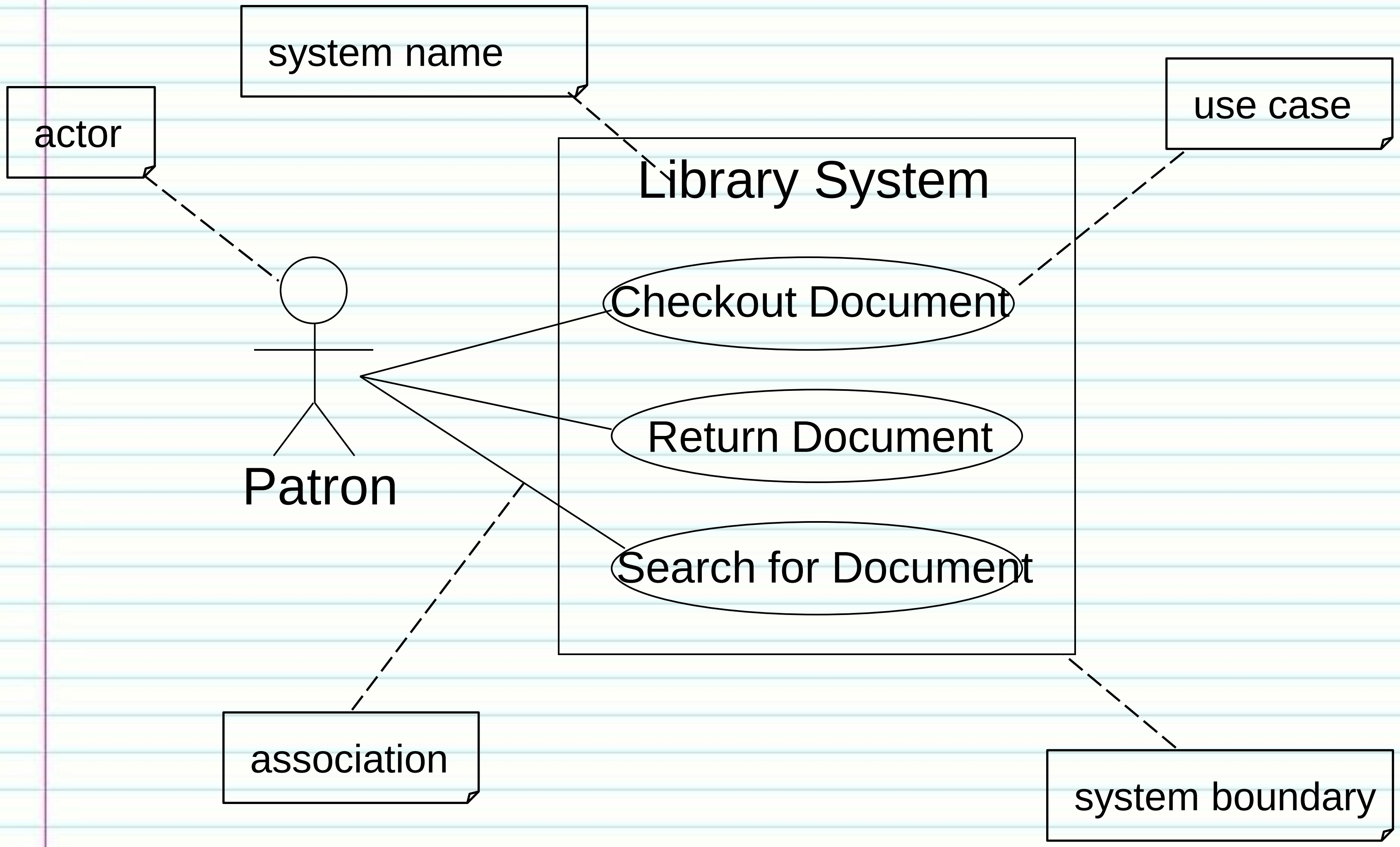
Use Case Diagram: Notion and Notation

Notion	Meaning	Notation
Use case	A use case is a business process that begins with an actor, ends with the actor, and accomplishes a business task useful for the actor.	
Actor	An actor is a role played by and on behalf of a set of business entities or stakeholders that are external to the system and interact with the system.	
System Boundary	It encloses the use cases and shows the capabilities of the system.	
Association between actors and use cases	It indicates that the actor uses the use case.	

Advanced Notions and Notations

Notion	Meaning	Notation
Inheritance	It indicates that one use case is more general/ specialized than the other.	——▷ Pointing from specialized use case to generalized use case.
Extension	It indicates that one use case can optionally continue the process of another use case.	– – – – – – –> <<extend>> Pointing from extended use case. Note: this is backwards from include
Inclusion	It indicates that one use case includes another use case as part of its business process.	– – – – – – – – – –> <<include>> Pointing from including use case to included use case.

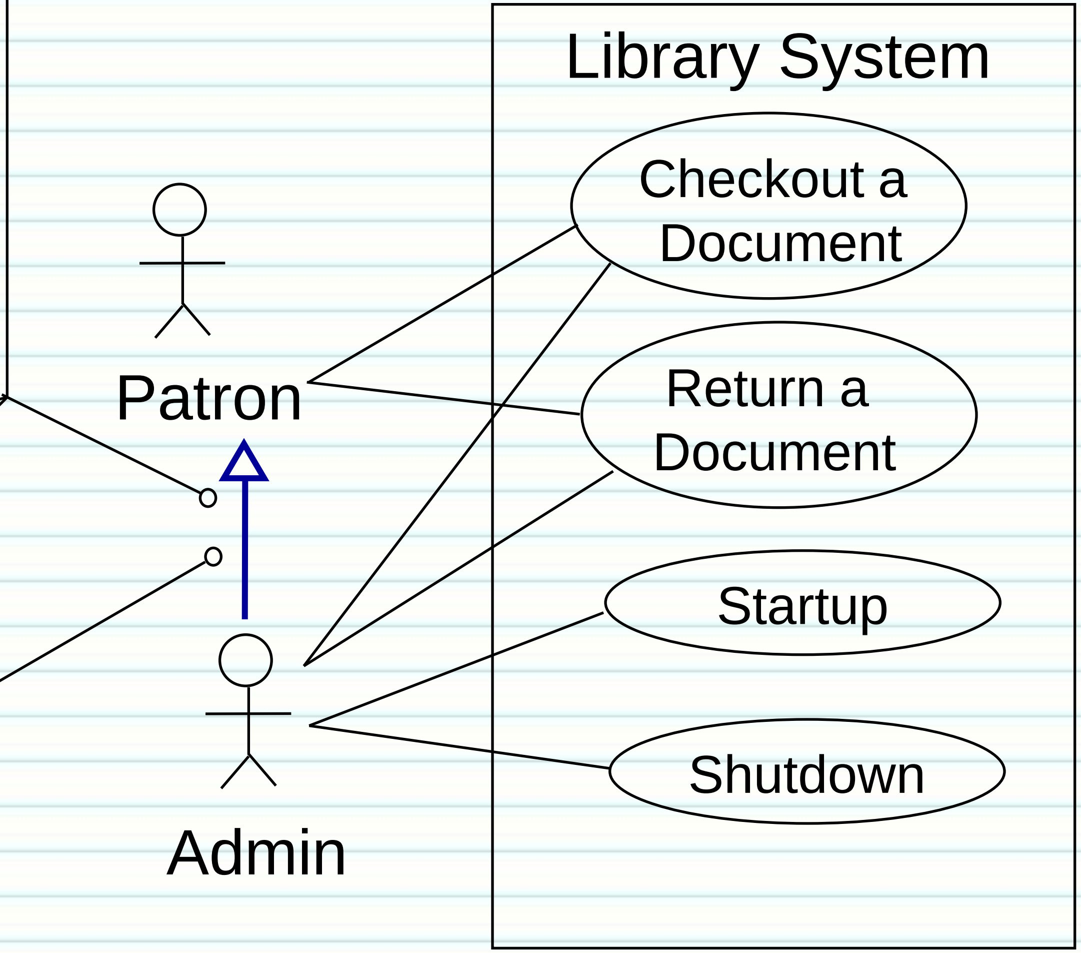
Use Case Diagram: Library Example



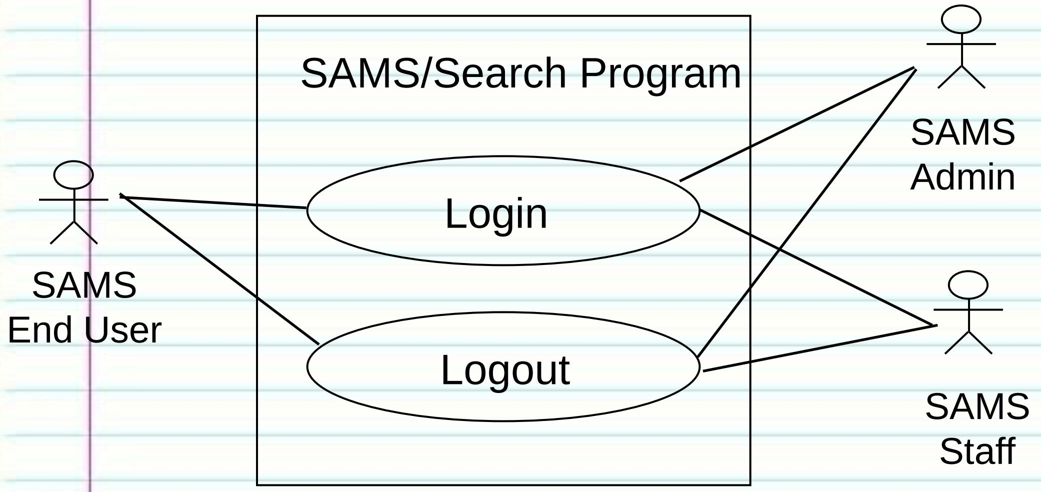
Simplify with Use of Inheritance

An admin can also checkout and return a document. But a patron cannot start or shutdown the system.

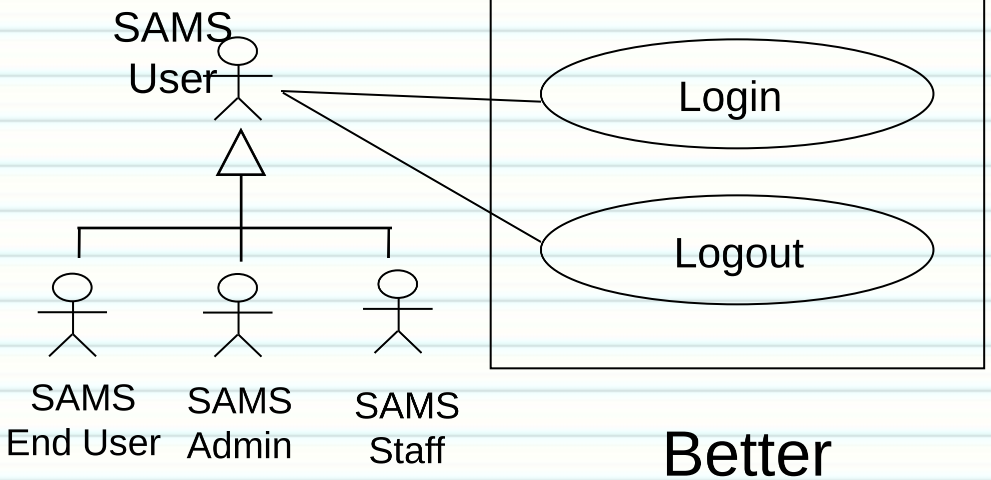
Inheritance: It states that an Admin IS-A Patron.



Simplify with Use of Inheritance



This is OK.



Better

Use Case Actor Inheritance Questions

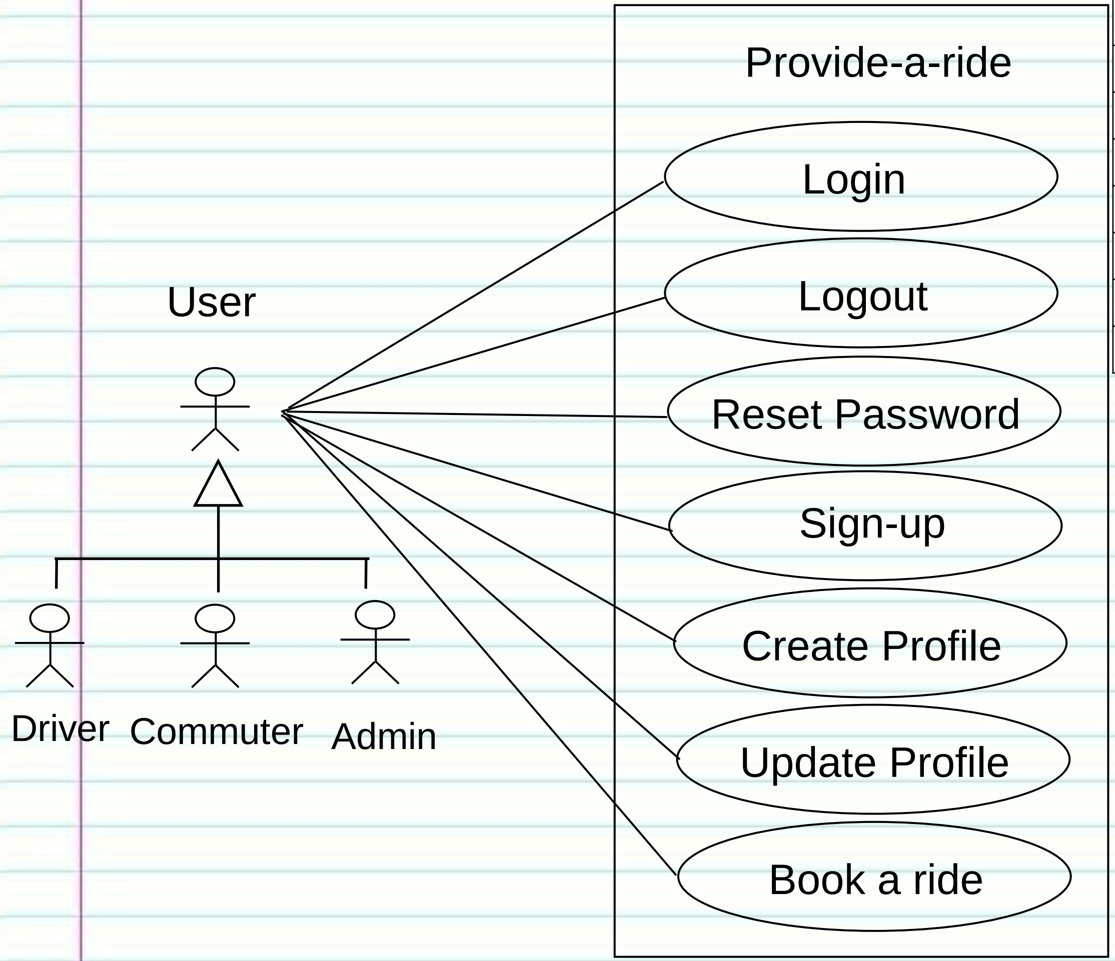
- Student exercise - let's say that we have the following usage table

	Usage by actor		
Use Case Name	**Driver**	**Commuter**	**Admin**
Login	X	X	X
Logout	X	X	X
Reset Password	X	X	X
Sign-up	X	X	
Create Profile	X	X	
Update Profile	X	X	
Book a ride	X	X	

- What is the best way to depict this? Develop your solution using pencil and paper. Make sure to use inheritance where needed

Use Case Actor Inheritance Questions (cont.)

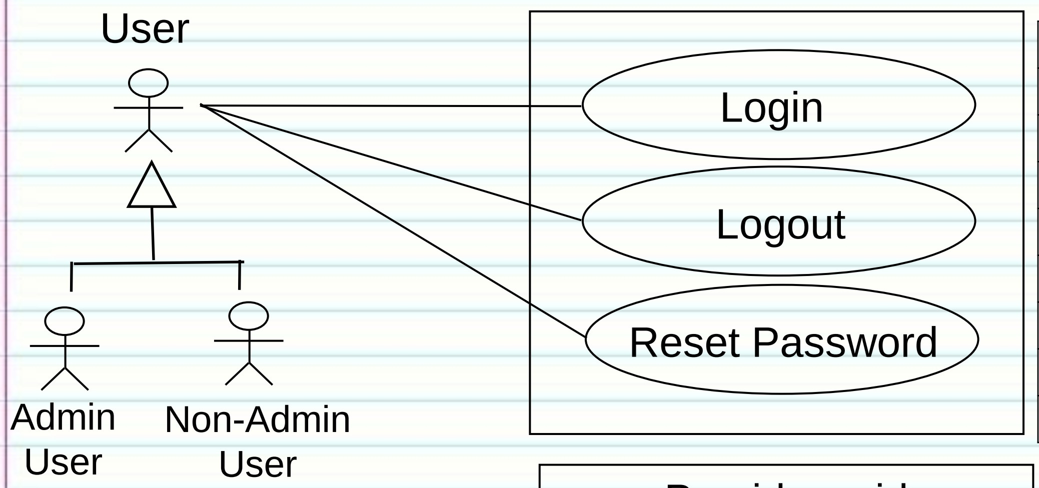
- What's wrong with the following diagram?



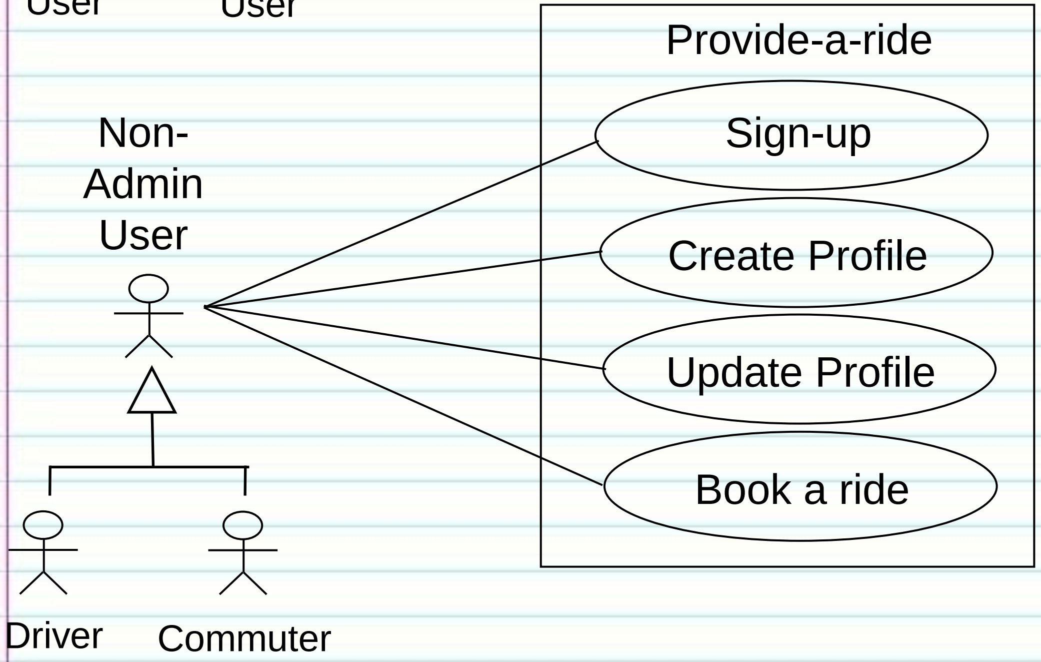
	Usage by actor		
Use Case Name	**Driver**	**Commuter**	**Admin**
Login	X	X	X
Logout	X	X	X
Reset Password	X	X	X
Sign-up	X	X	
Create Profile	X	X	
Update Profile	X	X	
Book a ride	X	X	

Use Case Actor Inheritance Questions (cont.)

- How about this solution?



	Usage by actor		
Use Case Name	**Driver**	**Commuter**	**Admin**
Login	X	X	X
Logout	X	X	X
Reset Password	X	X	X
Sign-up	X	X	
Create Profile	X	X	
Update Profile	X	X	
Book a ride	X	X	



Are the followings use cases? Why?

- Check authorization / check authentication
- Enter a password
- Process data
- Open a file
- Click on a menu item
- Traverse a linked list.
- Start a system

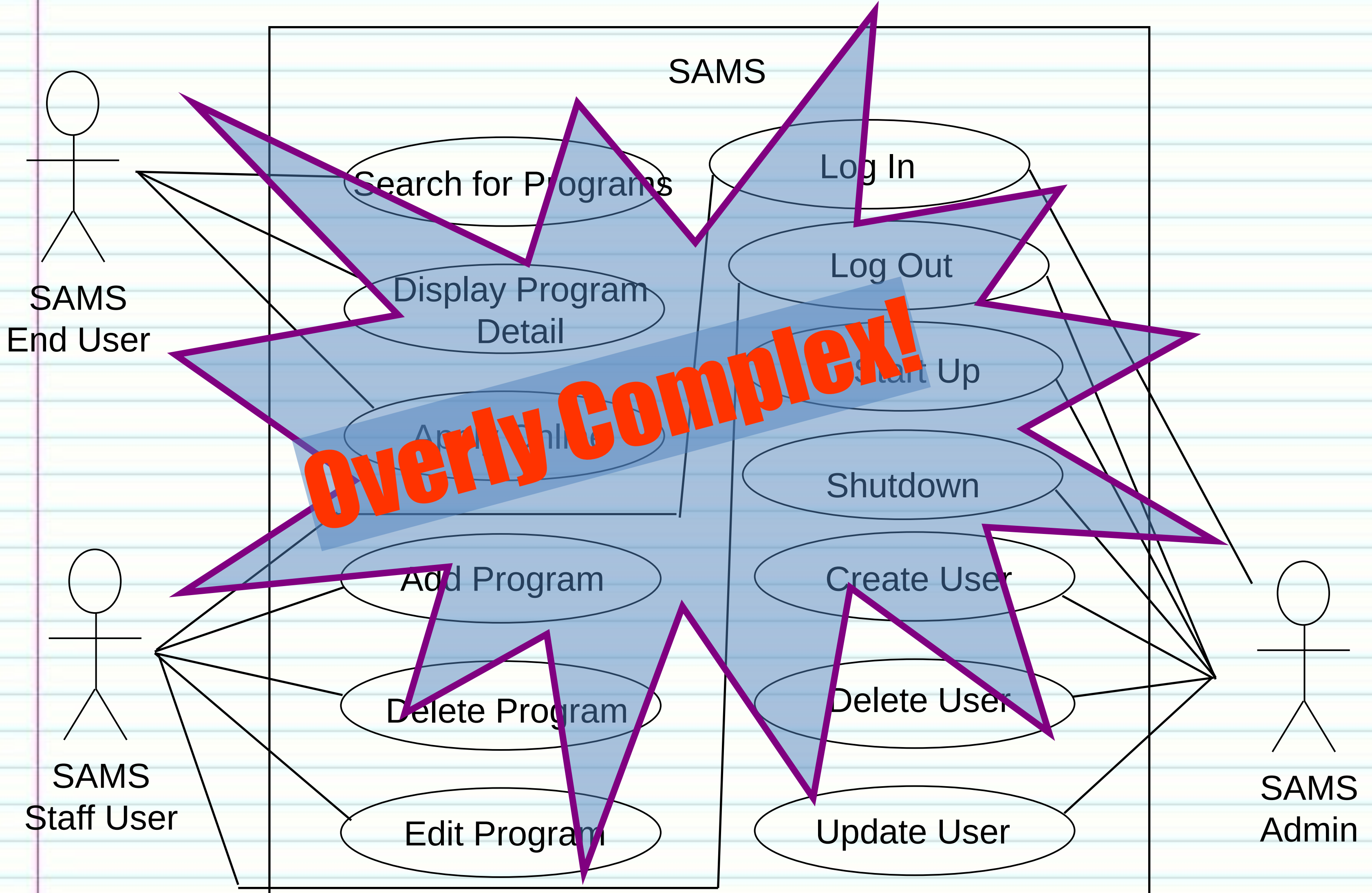
Guidelines for Use Case Diagram

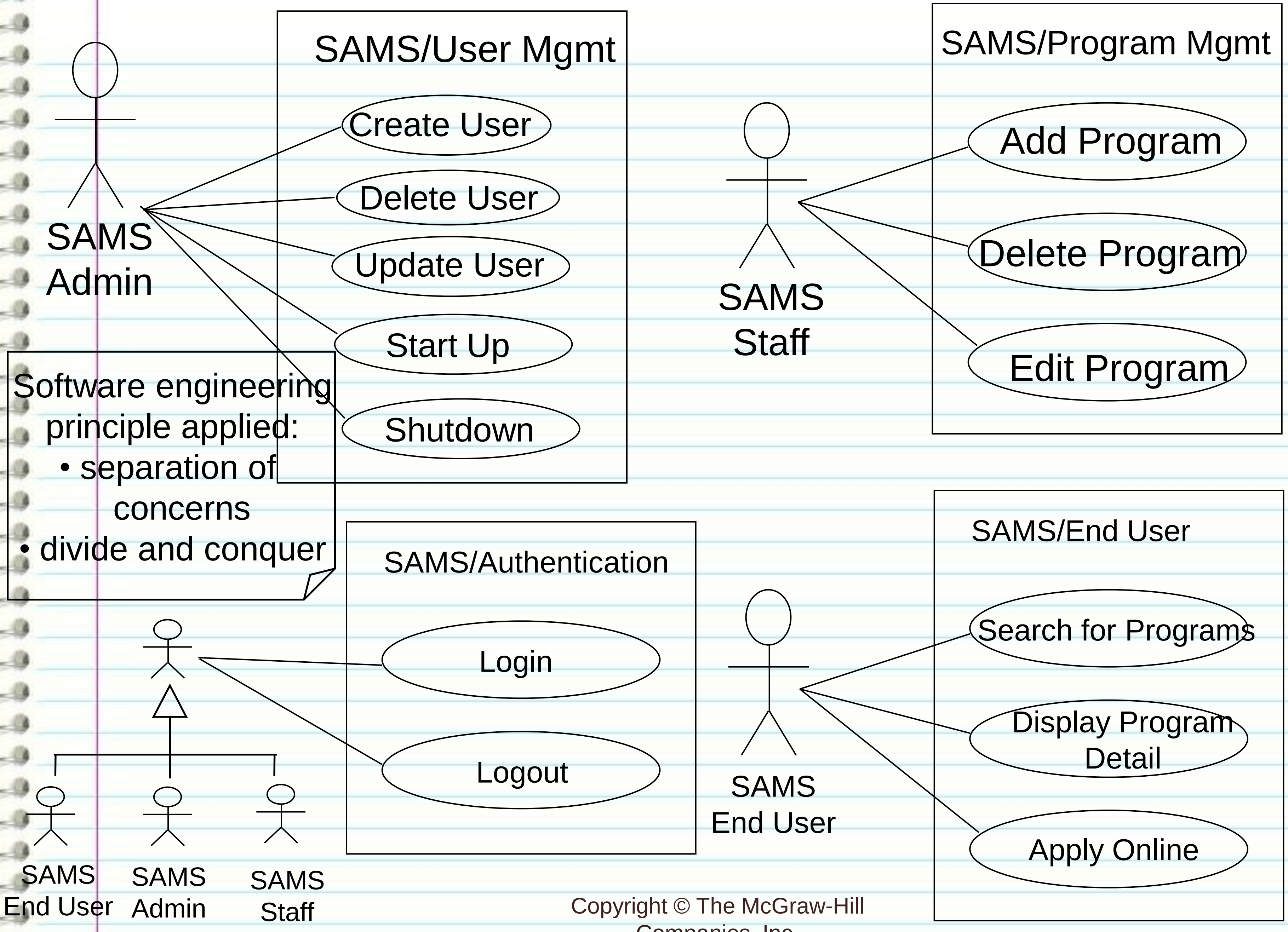
- Avoid showing
  - many use cases in one diagram (see next slide)
  - many use case diagrams each containing only one use case
  - overly complex use case diagrams
  - unnecessary relationships between use cases
- Use several diagrams to show groups of closely related use cases:
    - show only use cases and actors that are relevant
    - provide a meaningful name for the system/subsystem that implements group of use cases

Guidelines for Use Case Diagram

- Use inheritance to simplify the diagram by reducing the number of actor-use case links.
- Give a meaningful name for the system/subsystem that contains the use cases. The name may serve as the package or module name in design/implementation.
- Actor-use case relationships are always association relationships.
- Only use cases and their relationships can be shown within the system boundary.

Use Case Diagram

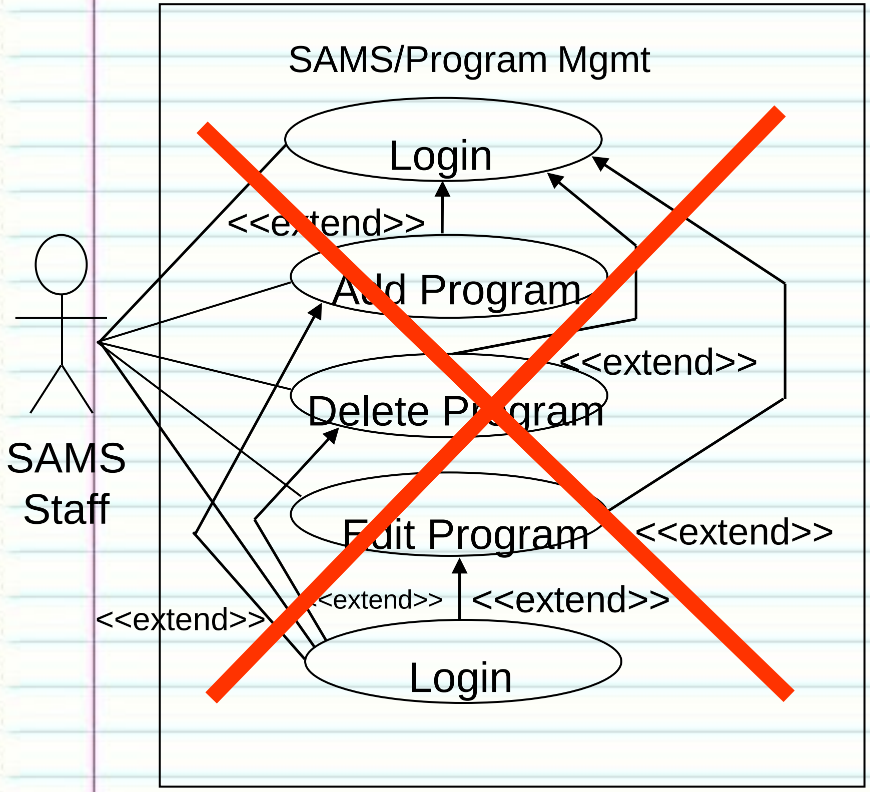


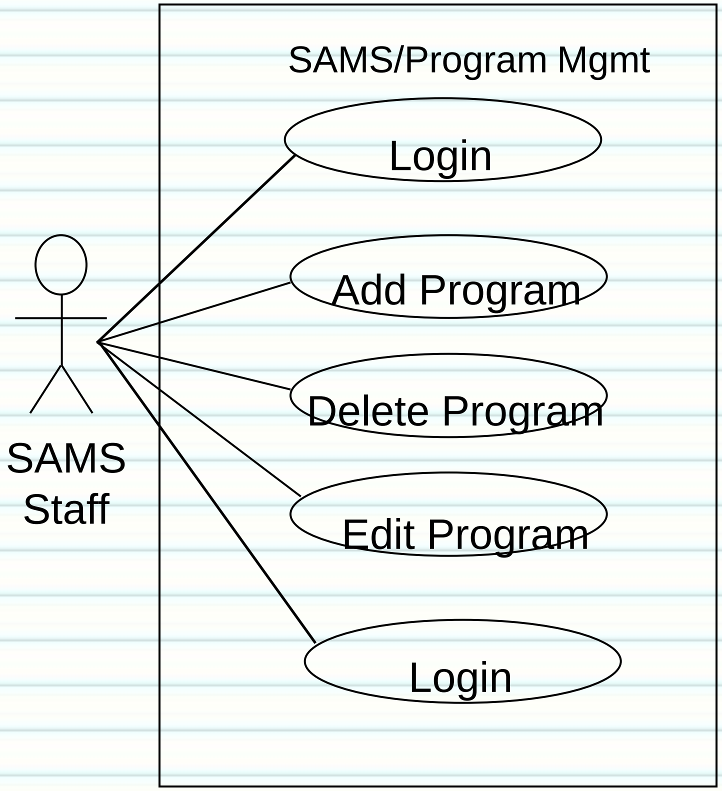


Software engineering principle applied:

- separation of concerns
- divide and conquer

Do Not Make It Unnecessarily Complex

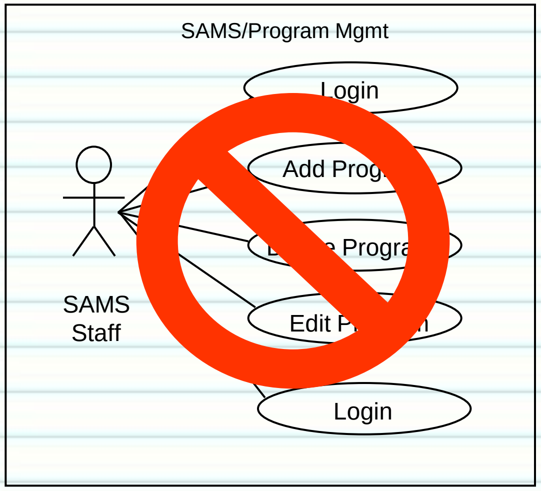


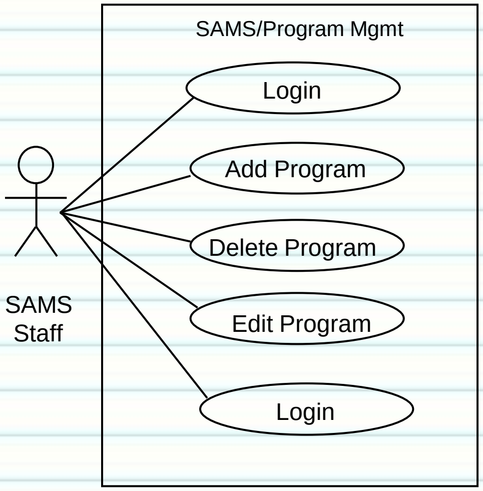


This diagram is made unnecessarily complex by adding the extend relationships.

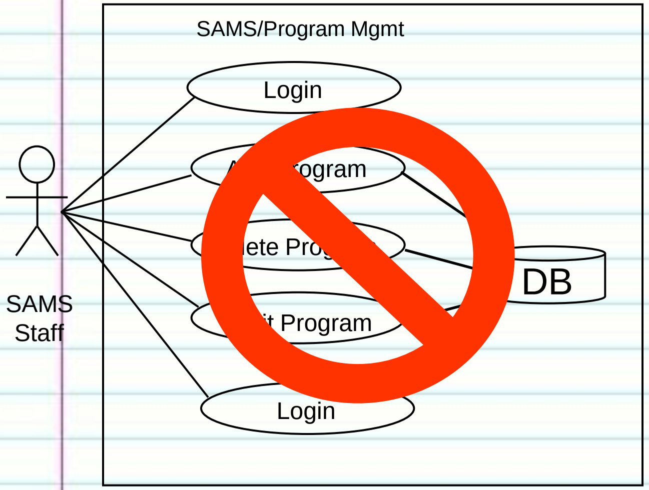
Much better

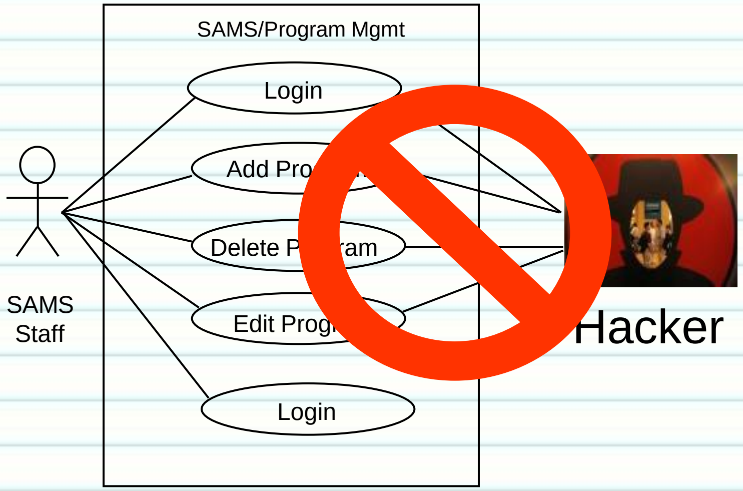
What Should Be In and Out?





Only use cases and their relationships are allowed in the boundary.





Use Case Modeling Steps

planning phase

iterative phase

requirements

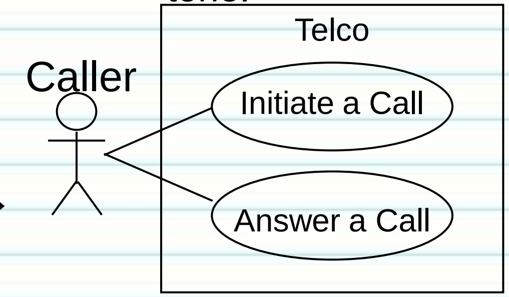
Deriving use cases from requirements

abstract use cases (e.g., Initiate a Call)

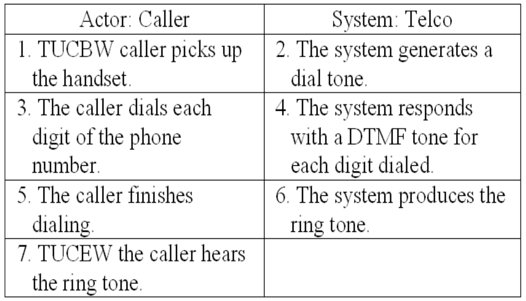
Defining use case scope

abstract & high level use cases

Example high level use case:
TUCBW caller picks handset from base
TUCEW caller hears the ring tone.



Depicting use case contexts



Actor: Caller	System: Telco
1. TUCBW caller picks up the handset.	2. The system generates a dial tone.
3. The caller dials each digit of the phone number.	4. The system responds with a DTMF tone for each digit dialed.
5. The caller finishes dialing.	6. The system produces the ring tone.
7. TUCEW the caller hears the ring tone.	

Specifying actor-system interaction (expanded use cases)

Deriving Use Cases from Requirements

- In the requirements specification, look for verb noun phrases that indicate *domain specific*
  - "do something"
  - "something must be done" or
  - "perform some task"

  in the application domain.
- Verify the verb noun phrases using use case definition (next slide).

Verify the Use Cases Identified

- Verify the use cases identified using use case definition:

  (1) Is it a business process? y/n

  (2) Is it initiated by an actor? y/n

  (3) Does it end with the actor? y/n

  (4) Does it accomplish something useful for the actor? y/n
- *All the answers to above questions must be "y."*

Identify Actor, System or Subsystem

- From the requirements, identify also
  - the actors, who initiate the tasks, or for whom the tasks are performed
  - the system or subsystem that contains the use case

Example: Library System

- Requirements of a library system:

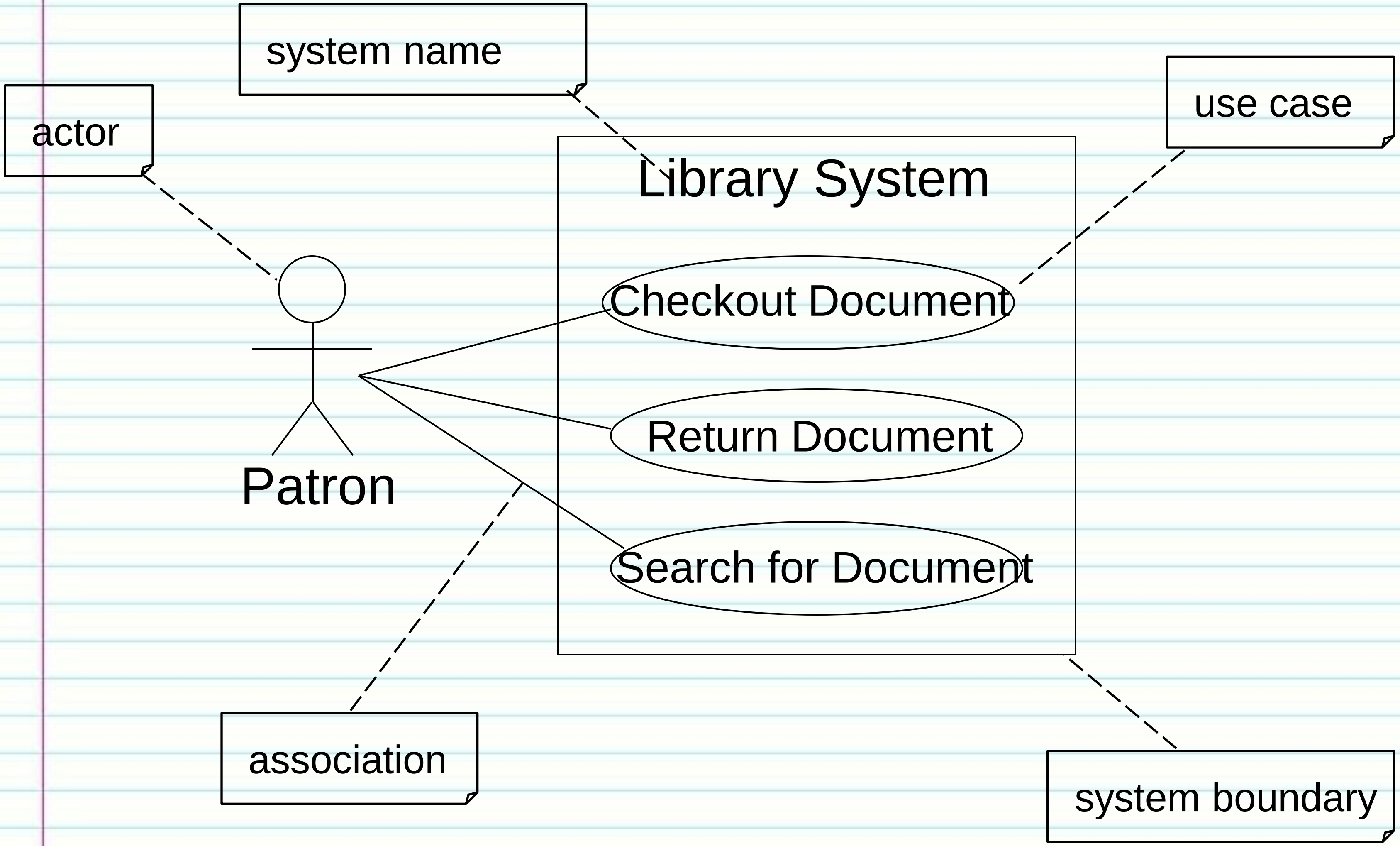
R1. The library system must allow a patron to check out documents.

R2. The library system must allow a patron to return documents.

discuss

	Use Case	Business Process?	Begin w/ Actor?	End w/ Actor?	Useful Task for Actor?	Use Case?	Actor	System
R1	Checkout Document	Y	Y	Y	Y	Y	Patron	Library System
R2	Return Document	Y	Y	Y	Y	Y	Patron	Library System

Use Case Diagram: Library Example



Example: Oversea Exchange Program

R1. The web-based application must provide a search capability for overseas exchange programs using a variety of search criteria.

R2. The web site must provide a hierarchical display of the search results to facilitate user navigation from a high level summary to details about an overseas exchange program.

	Use Case	Business Process?	Begin w/ Actor?	End w/ Actor?	Useful Task for Actor?	Use Case?	Actor	System
R1	Search for Programs	Y	Y	Y	Y	Y	User	Web App.
R2	Display Program Details	Y	Y	Y	Y	Y	User	Web App.

Business Process, Step, Operation and Action

- A business process is a series of steps to accomplish a complete business task.
- An operation is a series of actions or instructions to accomplish a step of a business process.

Business Process, Step, Operation and Action

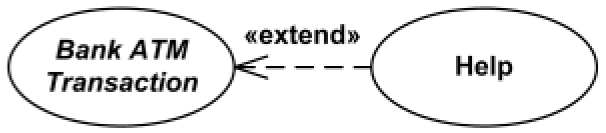
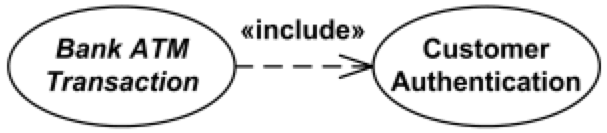
Application	Use Case	Steps/Operations	Actions
Text Editor	Edit Report	Open Report	click File, select Open, navigate to appropriate directory, select file, click OK button
		Make Changes	add, delete and modify texts and graphics (these are editor specific editing actions)
		Save Report	click File, select Save
		Exit Editor	click File, select Exit
Class Diagram Editor	Edit Diagram	Open Diagram	click File, select Open, navigate to appropriate directory, select file, click OK button
		Add Class	right click in canvas, select Add Class, fill in class information, click OK button
		Save Diagram	click File, select Save
		Exit Editor	click File, select Exit
ATM	Deposit Money / Withdraw Money	Start	insert card
		Authenticate	enter password, press Enter key
		Do transaction	select transaction type, enter deposit/withdraw amount, insert cash/take cash, take deposit/withdraw slip
		Finish	press Exit button, take ejected card

Are the followings use cases? Why?

- Check authorization / check authentication
- Enter a password
- Process data
- Open a file
- Click on a menu item
- Traverse a linked list.
- Start a system

Use Case Relationships

- Note the arrow direction on these

Extend	Include
	 Bank ATM Transaction «include» Customer Authentication
Base use case is complete (concrete) by itself, defined independently.	Base use case is incomplete
Extending use case is optional, supplementary.	Included use case required, not optional.

Use Case Specification: 3 Levels of Abstraction

- Abstract use case: a *verb-noun* phrase
- High level use case: *when and where* the use case begins and *when* it ends
  - TUCBW (This use case begins with ...)
  - TUCEW (This use case ends with ...)
- Expanded use case: *step-by-step* description of *how the actor interacts with the system* to carry out the business process

- Abstract use case: Initiate a Call
- High level use case:
  - Use Case: Initiate a Call
  - TUCBW the caller picks up the handset from the phone base.
  - TUCEW the caller hears the ring tone.

High Level Use Case Example

Use Case: *Withdraw Money* (from an ATM)

TUCBW the ATM user inserts an ATM card into the card slot.
TUCEW the ATM user receives the correct amount of cash and a withdraw slip.

Use Case: *Search for Programs*

TUCBW a SAMS user selects the ``Search for Programs'' function from his homescreen.
TUCEW the user sees a list of programs satisfying the search criteria (“search results screen”) and may navigate back to his homescreen.

Guidelines for High Level Use Case

- A high level use case should not specify background processing:
  - Do not specify how the system process the request such as update the database.
  - Background processing is modeled by sequence diagrams.
- High level use cases should end with what the actor wants to accomplish.

Requirements Use Case Traceability

- Traceability attempts to address the following:
  - How do you know that the system will deliver all the capabilities stated in the requirements?
  - How do you know to what extent the system will satisfy the requirements?
  - How do you know if some use cases are missing?
  - How do you know which use cases are not needed?
  - How do you know which requirements are more important than the other?
  - How do you know which use cases should have high priority?

This is for Knowledge sakes only - we will not use the RUTM for the Project but I do have a question on the quiz

Requirements-Use Case Traceability Matrix

	Priority Weight	UC1	UC2	UC3	UC4	UC5	UC6
R1	3	X	X				
R2	2					X	
R3	2	X					
R4	1		X	X			
R5	1				X		X
R6	1		X			X	
Score		5	5	1	1	3	1

Usefulness of the Traceability Matrix

- It highlights which use cases relate to which requirements, and vice versa.
- It shows the priorities of the requirements and use cases.
- Projects should focus on timely delivery of high-priority use cases.
- It is useful for use case based acceptance testing – high-priority use cases should be tested first.

Project Planning by Use Cases

1) Identify dependencies between the use cases:
   - use cases are business processes
   - business processes depend on each other: process P2 is impossible unless process P1 had been performed.
   - example: cannot return a book unless it had been checked out.

2) Compute a partial order to develop the use cases according to the dependencies.

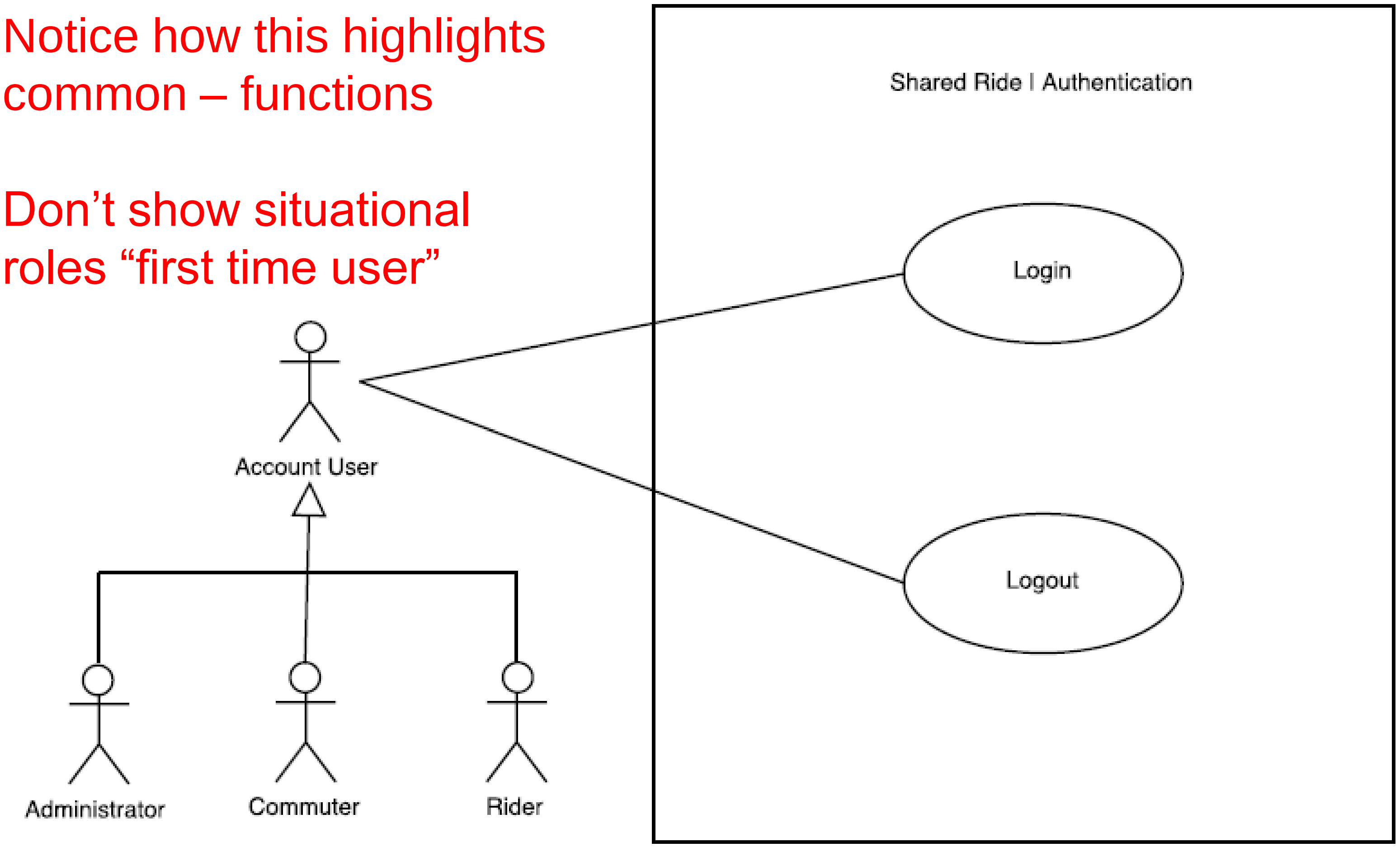
3) Schedule the development according to the partial order, favor higher priority use cases when no ordering exists between two use cases.

Example Project
Iteration 1 Materials

Use Case Diagrams

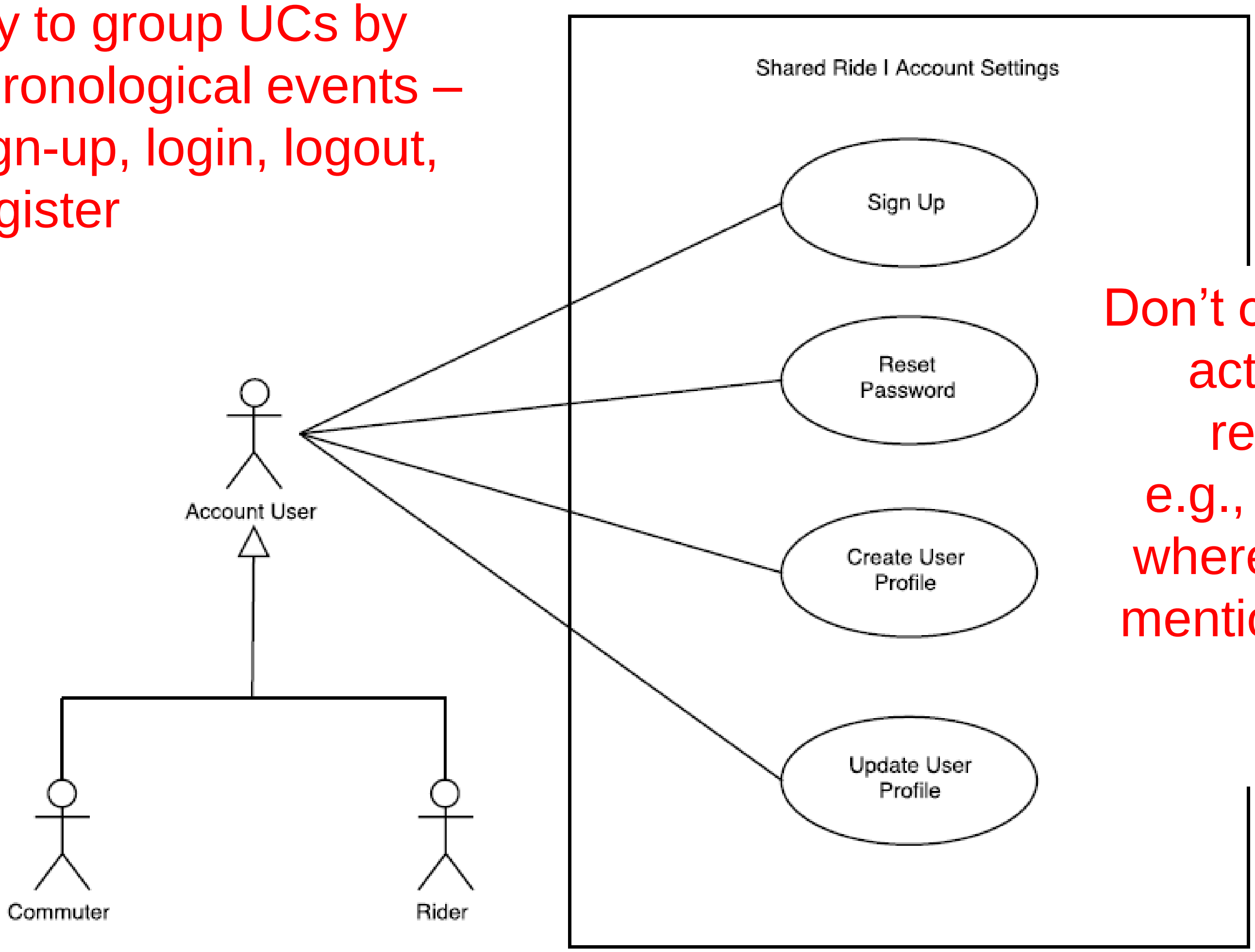
Notice how this highlights common – functions

Don’t show situational roles “first time user”



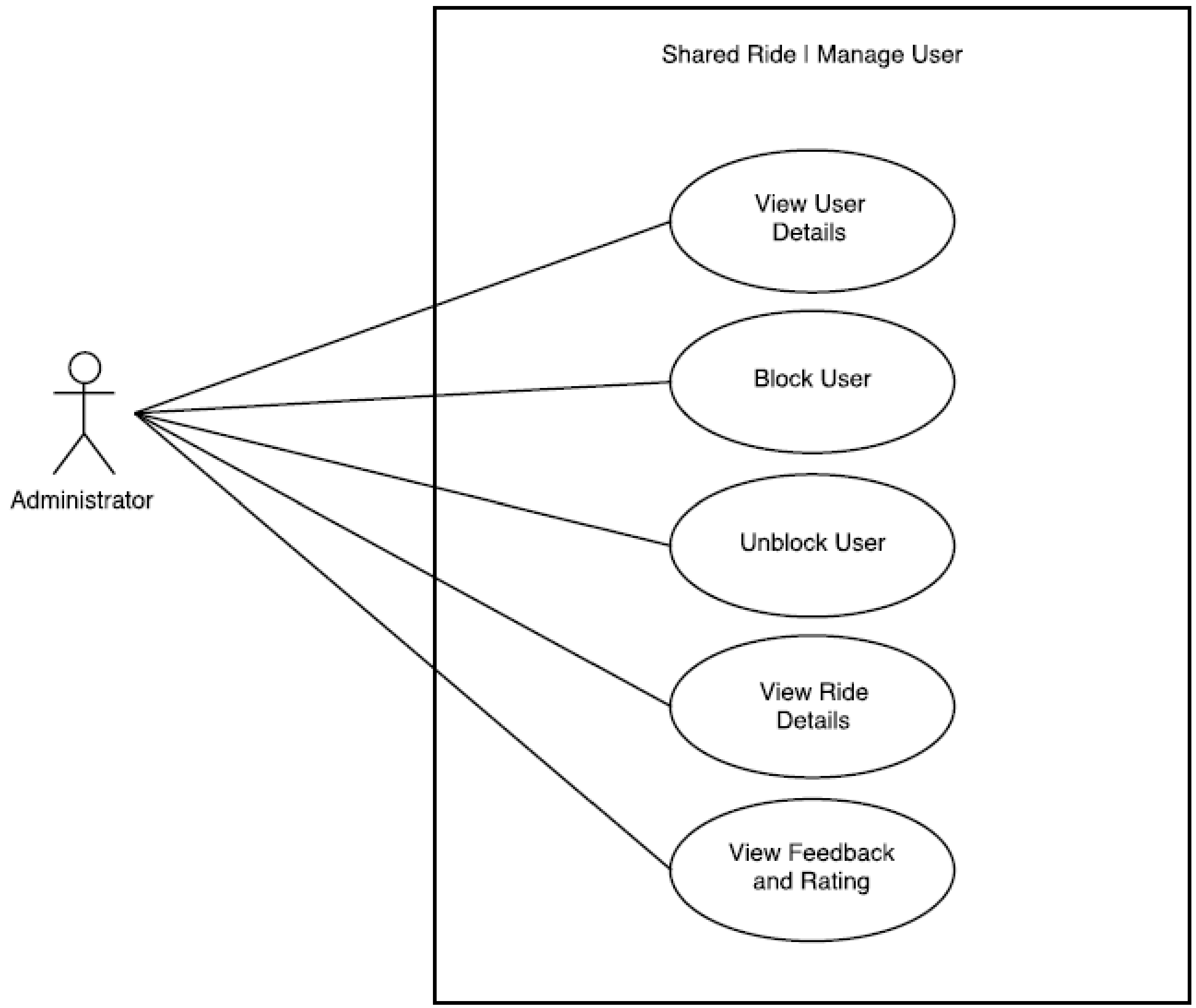
Use Case Diagrams (cont.)

Try to group UCs by chronological events – sign-up, login, logout, register

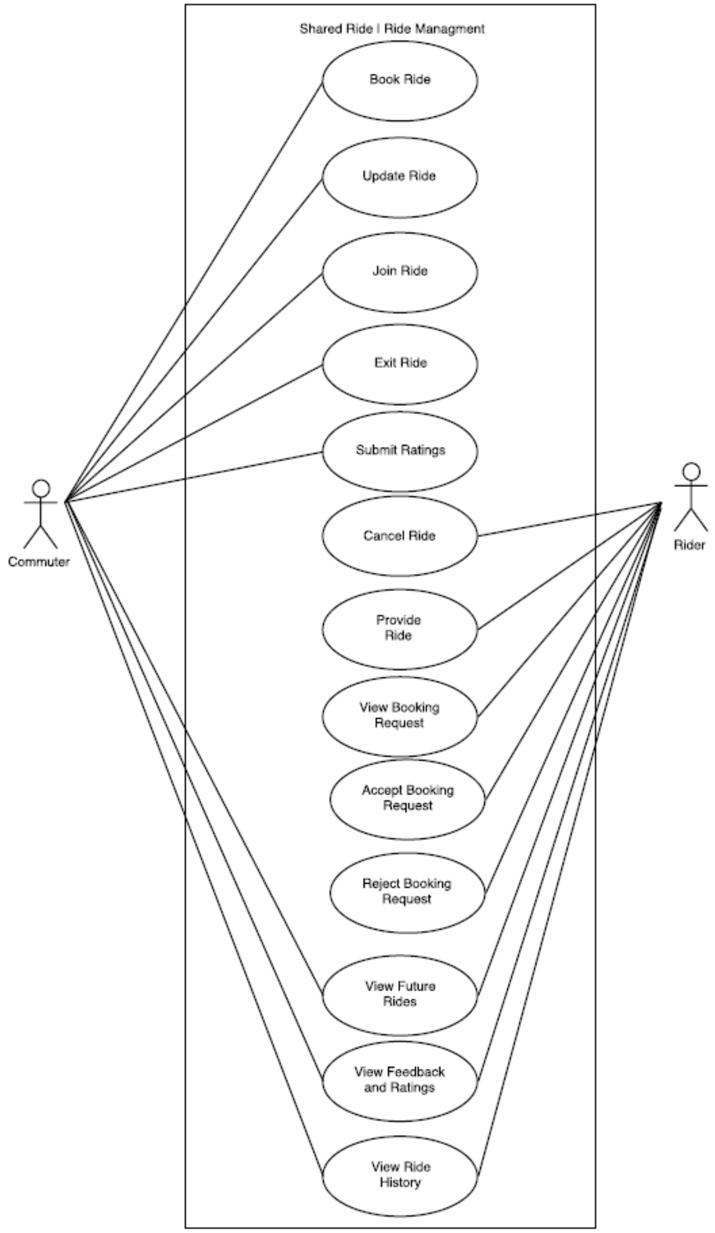


Don’t create a hierarchy of actors that is never referenced later - e.g., Non-admin User - where no other diagram mentions this Non-admin user

Use Case Diagrams (cont.)

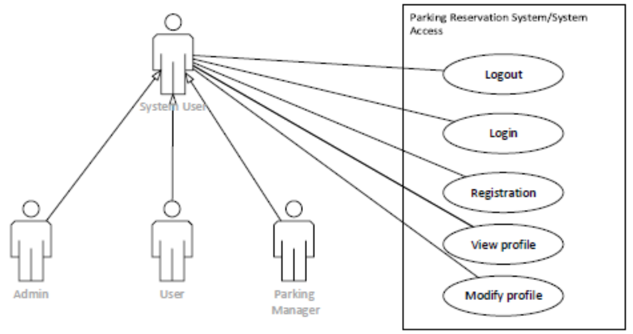


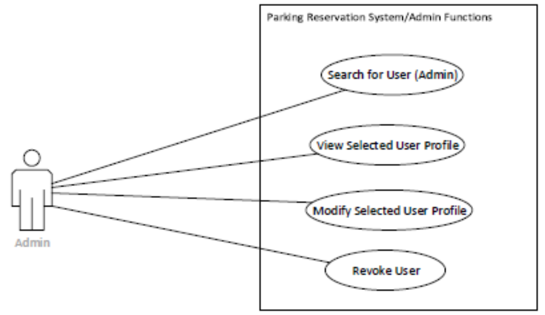
Use Case Diagrams (cont.)



Break up into separate diagrams to better show similar and different

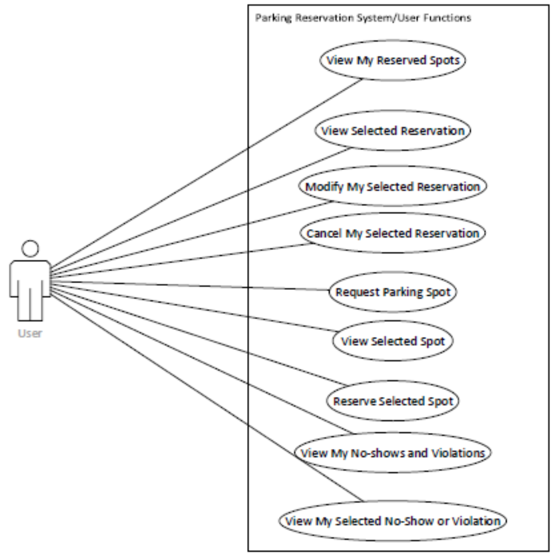
A Correct Use Case Diagram

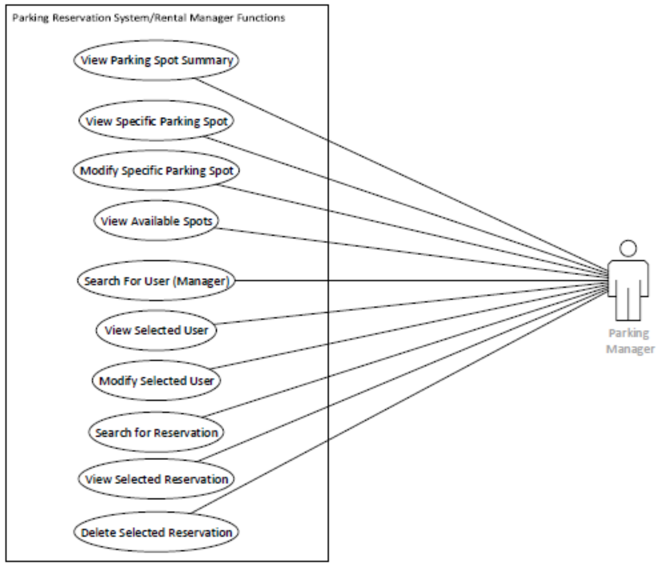



Parking Reservation System/Admin Functions
Search for User (Admin)
View Selected User Profile
Modify Selected User Profile
Revoke User
Admin

Excluding Login/Logout/etc create one diagram per system user

1) Unlike the previous semester's solution there is no need for a non-admin user as all have the same function access


Parking Reservation System/User Functions
View My Reserved Spots
View Selected Reservation
Modify My Selected Reservation
Cancel My Selected Reservation
Request Parking Spot
View Selected Spot
Reserve Selected Spot
View My No-shows and Violations
View My Selected No-Show or Violation
User


Parking Reservation System/Rental Manager Functions
View Parking Spot Summary
View Specific Parking Spot
Modify Specific Parking Spot
View Available Spots
Search For User (Manager)
View Selected User
Modify Selected User
Search for Reservation
View Selected Reservation
Delete Selected Reservation
Parking Manager

UCD and the Class Project

- Start with the UCID - there should be a one-to-one mapping between UCID functions and the UCs in the UCD
- Make sure to get system users correct between the two - let the UCID drive the UCD - the UCD should simply be a visual representative of the UCID
- Use actor inheritance where needed
- Show common/unique UCs at a glance