UML 2.0

Use-Case Diagrams

UML 2.0

- Class diagrams (+ OCL constraints)
- Package diagrams
- Component diagrams
- Deployment diagrams
- Use cases diagrams
- State diagrams
- Activity diagrams
- Interaction diagrams

UML 2.0

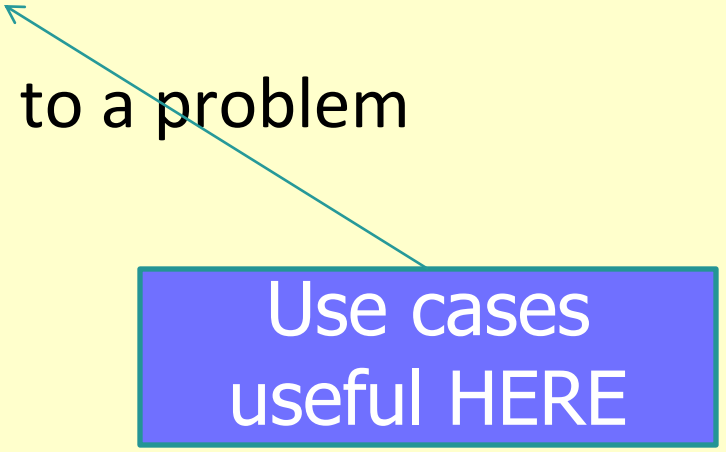
- ✓ Class diagrams (+ OCL constraints)
- Package diagrams
- Component diagrams
- Deployment diagrams
- Use case diagrams : today !
- State diagrams
- Activity diagrams
- Interaction diagrams

The Design of a Software Application

- Collecting requirements (*expression des besoins*)
- Specifications (*cahier des charg*es)
- Analysis: problem description
- Design: providing the solution to a problem
- Implementation and unit test
- Integration test
- Final tests (*recettes*)

The Design of a Software Application

- Collecting requirements (*expression des besoins*)
- Specifications (*cahier des charges*)
- Analysis: problem description
- Design: providing the solution to a problem
- Implementation and unit test
- Integration test
- Final tests (*recettes*)



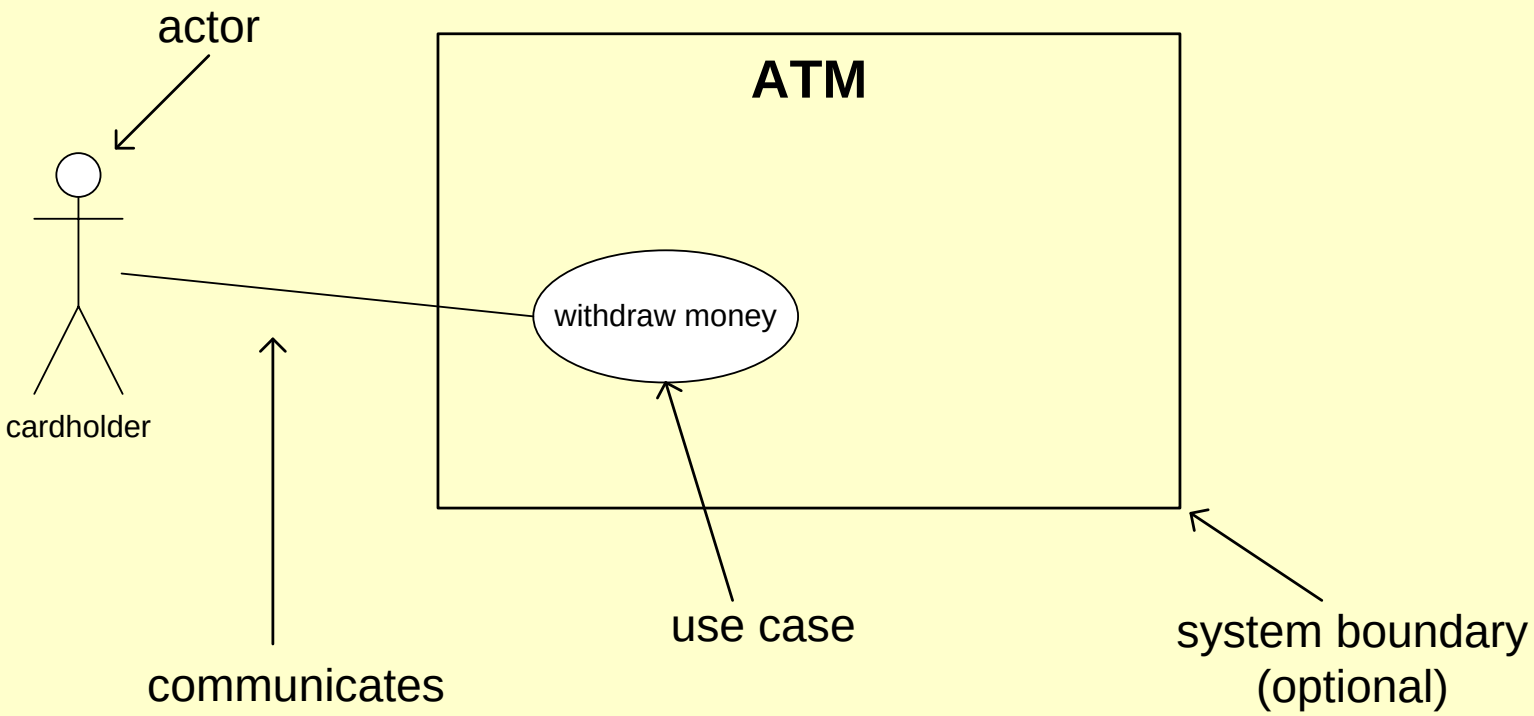
What is a Use Case?

- Simple answer: it's a reason to use a system
- Formal answer: it describes a functional requirement of a system as a whole from an external perspective
- Examples:
  - Library use case: **borrow a book**
  - ATM use case: **withdraw money**
  - Alarm clock use case: **set timer**
  - IT Help Desk use case: **log issue**

Use Cases in UML 2.0

- To describe a use case we have to identify:
  - the actors: types of user that interacts with the system
  - the system
  - the goal: the reason for the user to use the system
- Example:
  - a bank cardholder uses an ATM machine to withdraw money

Use Case Diagrams in UML 2.0



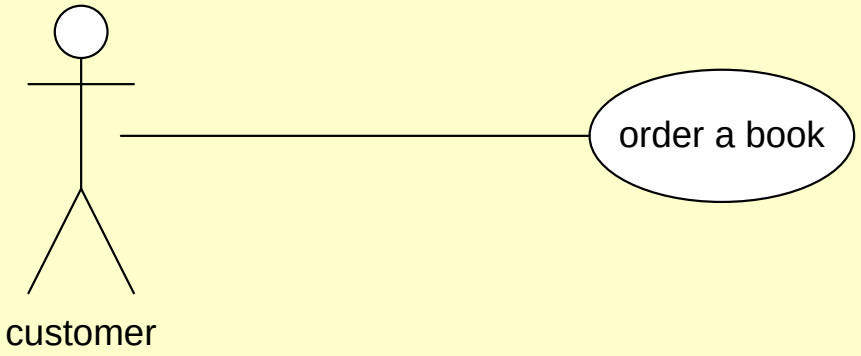
Actors

- Actors describe external roles
- Actors initiate (and respond to) use cases
  - *Driver* starts car
  - *Cardholder* puts the card into the ATM machine
  - *User* sets the time on the DVD
  - *Timer* trigger email alert

Actors (2)

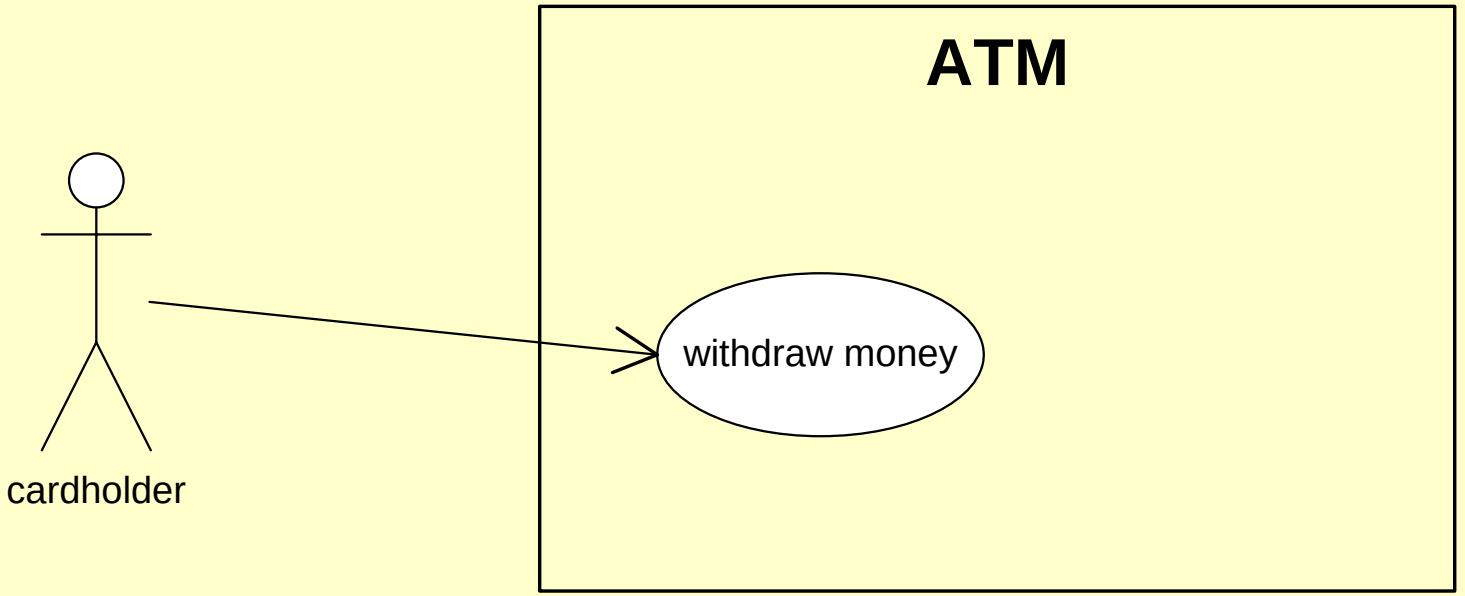
- The actor describes a role that the user play in relation to the system:
  - A cardholder can be an IT manager but that does not interest us!
- The actor is external to the system itself (again!)
- Actors are not necessarily people!
  - They can be people, pieces of software, or other systems…
- The goal must be of value to the actor:
  - A use case “cardholder enters PIN” makes no sense
  - We don’t build ATM machines for people to enter PINs!

Use Cases Diagrams in UML 2.0 (2)



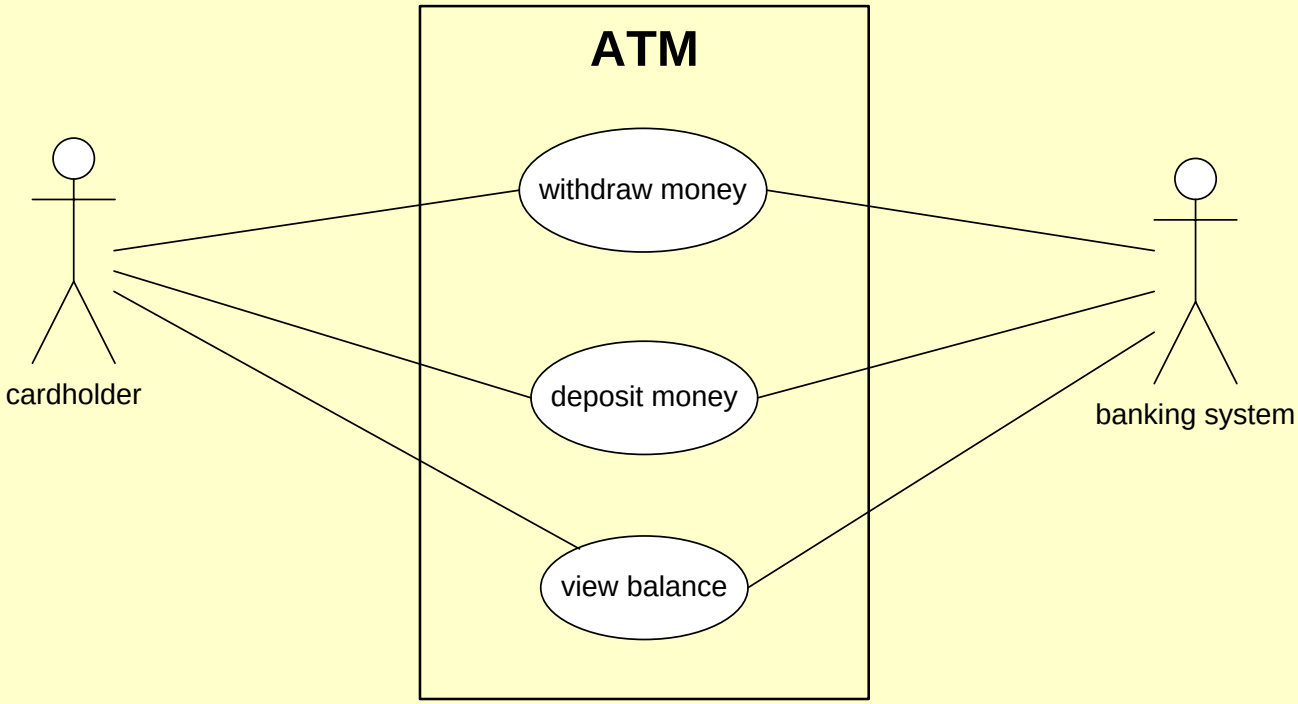
- A relationship between an actor and a use case can mean that:
  - the actor triggers the use case
  - the use case sends results to the actor
  - both

Use Cases Diagrams in UML 2.0 (3)



Optional: an arrow indicates who always originates the transaction

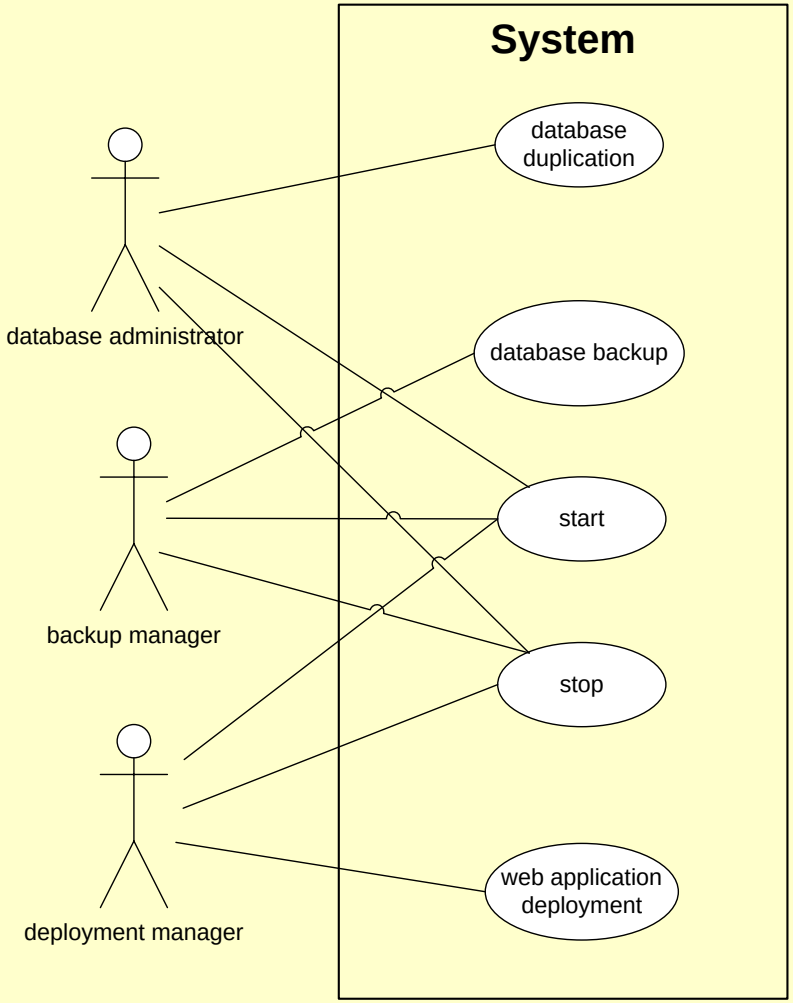
Actors and System Boundaries



Actors and Roles

There can be as many actors as roles

Jason can be the database administrator AND the backup Manager AND the depl. Manager

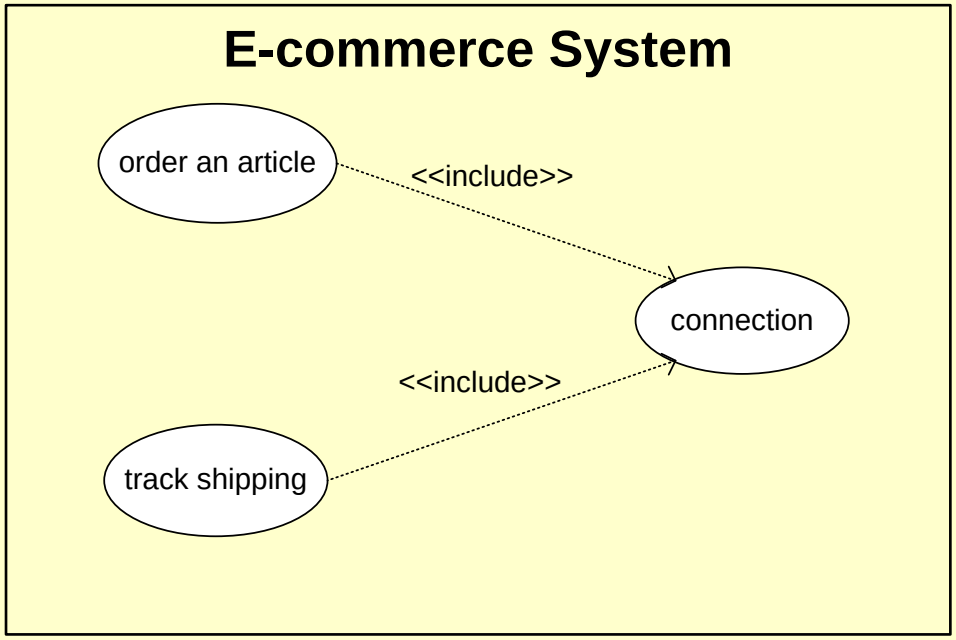


Relationships between Use Cases

- Includes
  - Ex. “go to work” *includes* “board a train”
- Extends
  - Ex. Is trains aren’t running, “catch a bus” may *extend* “go to work”
- Generalization
  - Ex. “Feed an animal” is a *generalization* of “feed a cat”

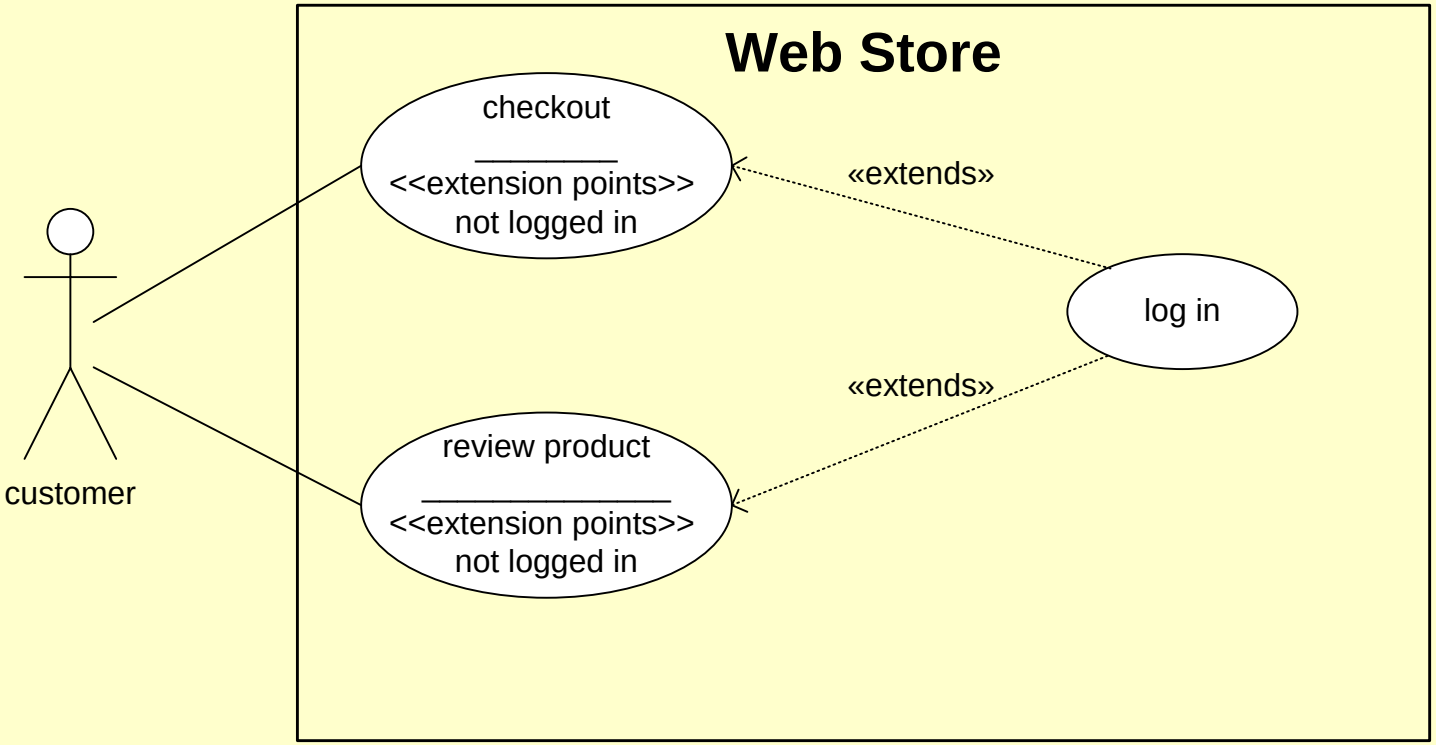
Include

- *A «includes» B*: it means that the execution of a use case A always includes the execution of use case B from a certain point onward
- Allows B to be “reused” by different use cases
- Use case B is neither complete nor autonomous!



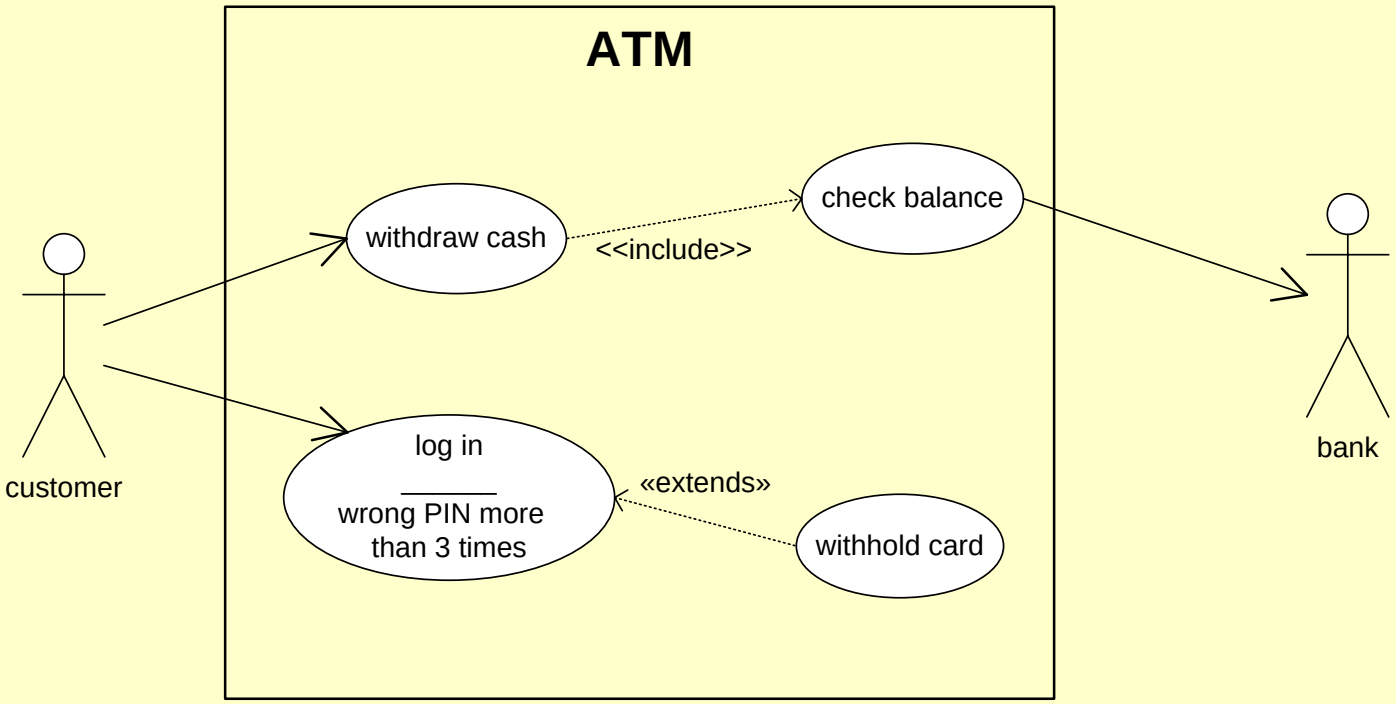
Extend

- Adds functionalities if certain conditions are met
- Conditions are specified through extension points

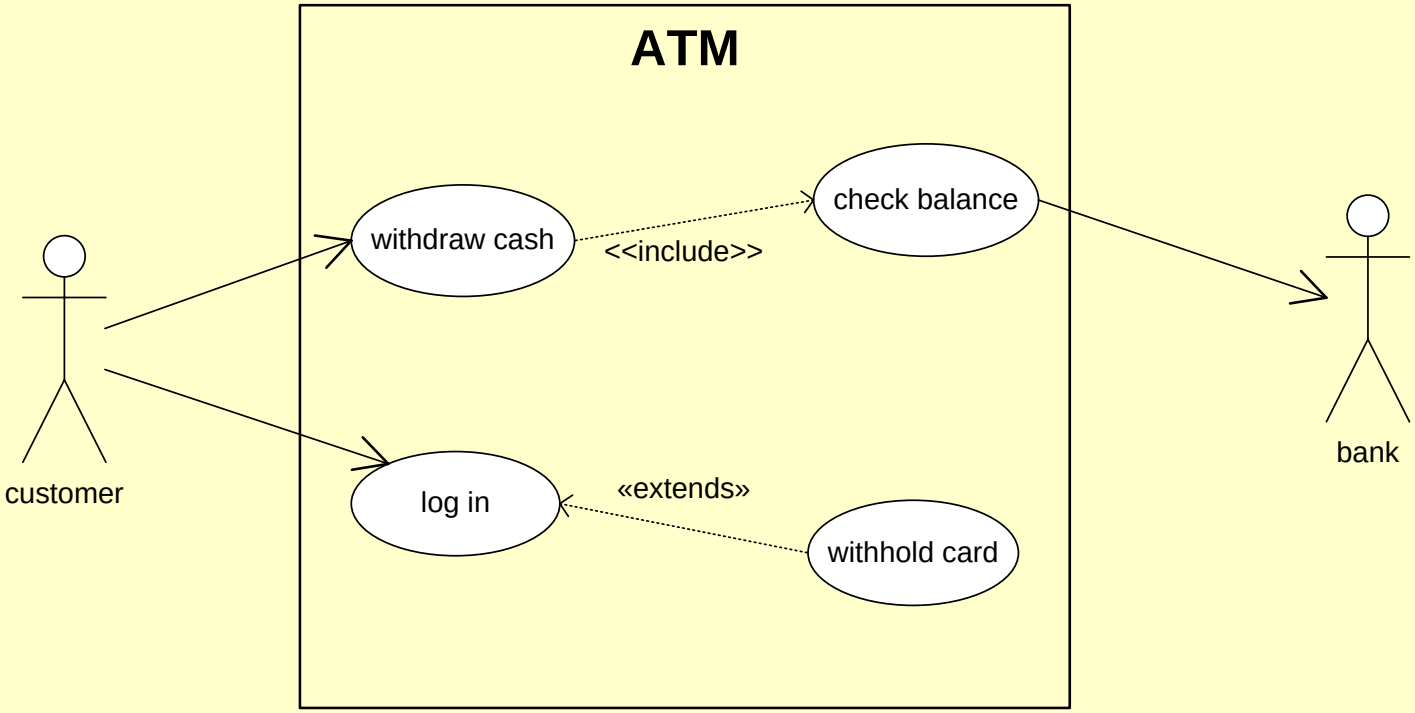


Include vs. Extend

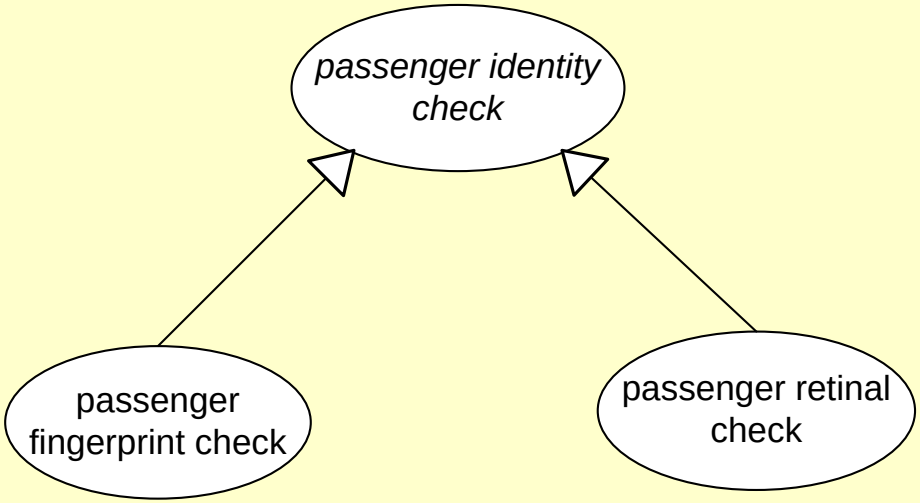
- «include» means always included
- «extend» means conditionally extended



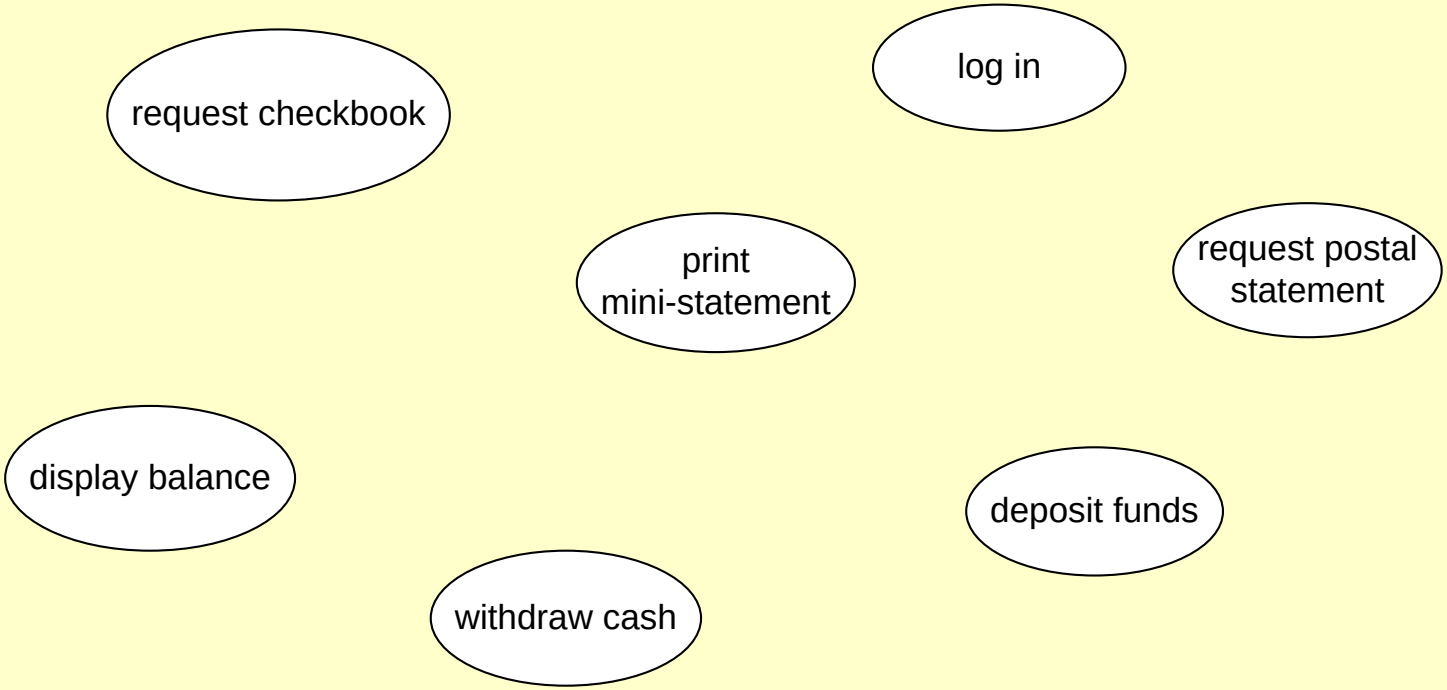
Example: Include and Extend



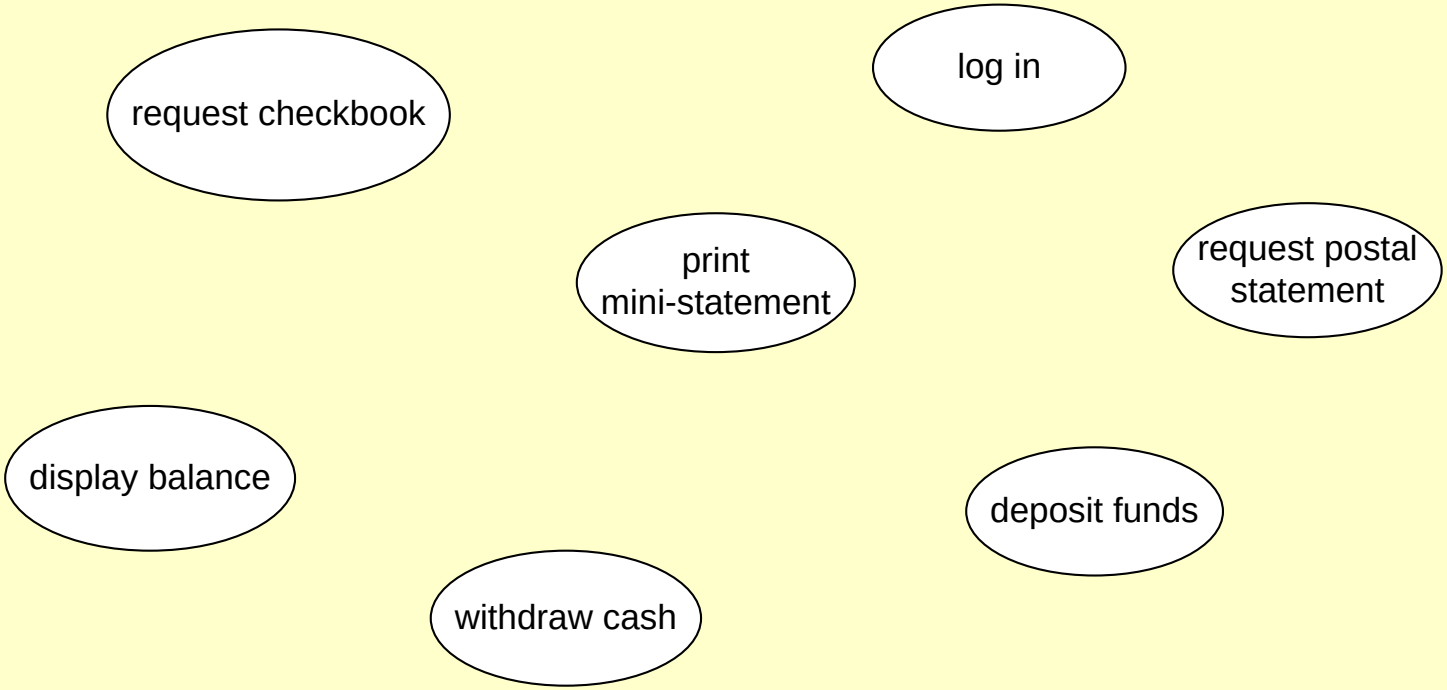
Generalization



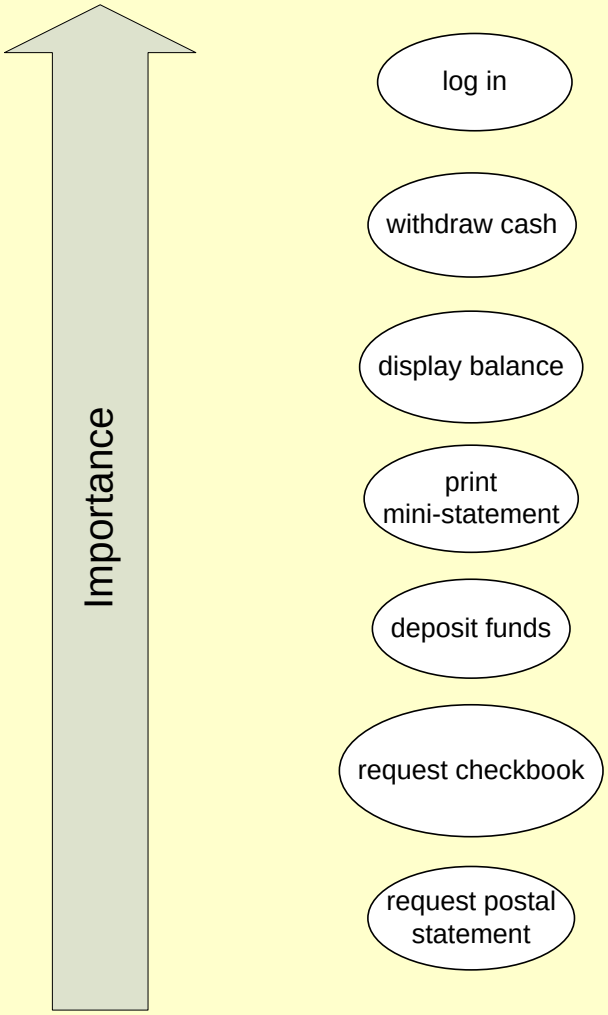
Use Case Driven Development



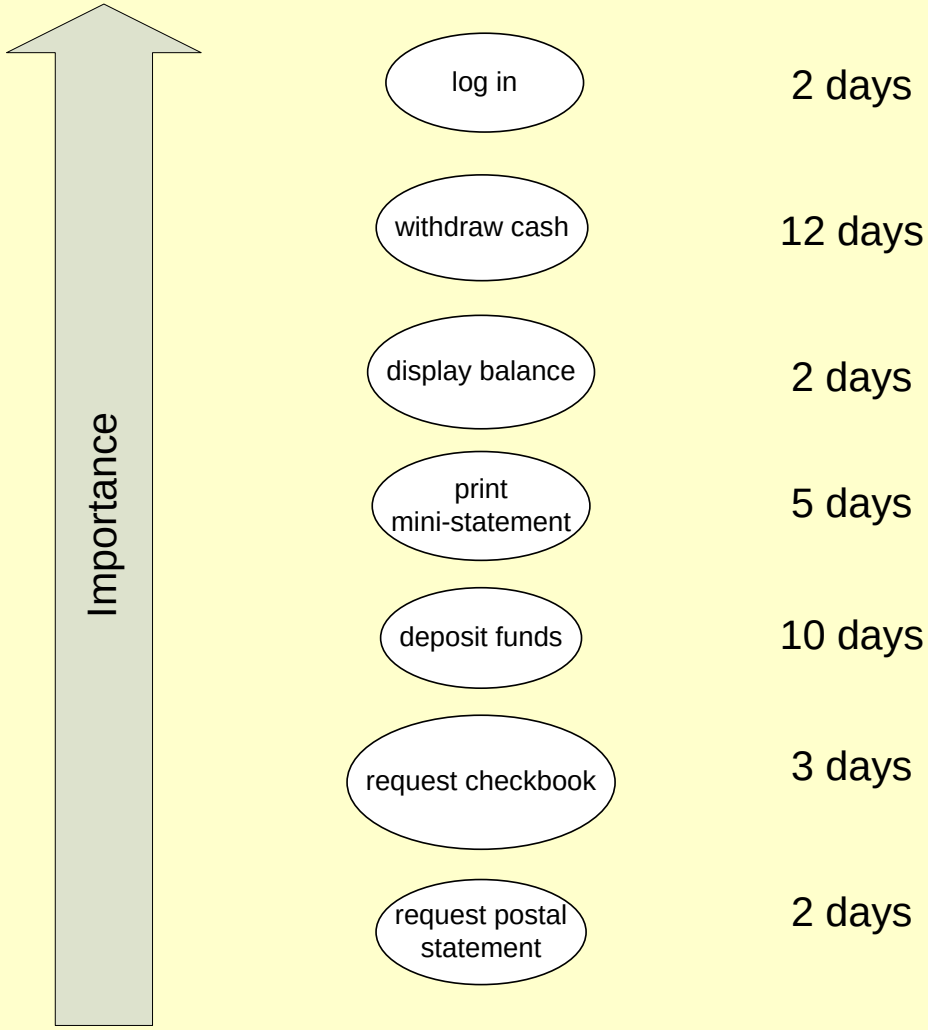
Use Case Driven Development



Use Case Driven Development



Use Case Driven Development



Use Cases: Common Pitfalls

- System boundary is undefined or inconstant
- Use cases are written from the system's (not the actors') point of view
- Actor names are inconsistent
- There are too many use cases
- Actor-to-use case relationships resemble a spider's web
- Use-case specifications are too long/confusing
- Use case doesn't correctly describe functional entitlement.
- Customer doesn't understand the use cases.
- Use cases are never finished.

Use Cases: a Simple Example

USE CASE: Place order
GOAL: To submit an order and make a payment
ACTORS: Customer, Accounting
PRIMARY FLOW:

1. Customer selects “Place Order”
2. Customer enters name
3. Customer enters product codes for products to be ordered.
4. System supplies a description and price for each product
5. System keeps a running total of items ordered
6. Customer enters payment information
7. Customer submits order
8. System verifies information, saves order as pending, and forward information to accounting
9. When payment is confirmed, order is marked as confirmed, and an order ID is returned to customer

Use Case : fiche descriptive

USE CASE # 1	*Forme verbale (infinitif)*		*(ex : Passer une commande)*
Résumé			
Acteur principal	X		
Intervenants & Intérêts	*Intervenants*	*Intérêt*	
	X Y Z		
Préconditions			
Garanties de succès	*(Conditions d'arrivée)*		
Déclencheur	*(Evènement)*		
Scénario nominal	Etape	Action	*(Le cas où tout se passe bien)*
	1	L'utilisateur	
	2	X lance	
	3	Le système	
	4	X lance le processus (Cas d'utilisation 1.1)	
	5	Le système.	
Extensions	Etape	Action	*(Variations au scénario et cas d'erreurs)*
	2a	X reçoit ...	
		2a.1 X fait ceci	
		2a.2 Y fait cela	