Use Case Relationships

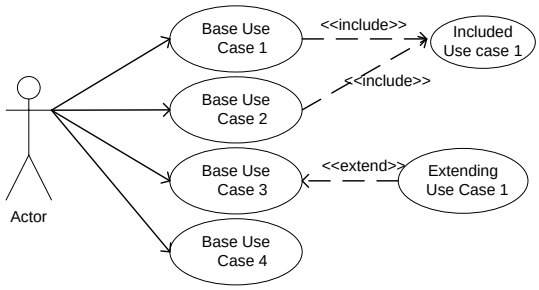
Il-Yeol Song, Ph.D.
Professor
College of Information Science and Technology
Drexel University
Philadelphia, PA 19104
Song@drexel.edu
http://www.ischool.drexel.edu/faculty/song/

Objectives

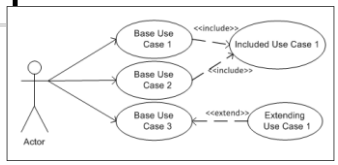
- Modeling complex relationships between/among use cases: Learn to employ Use Case Relationships
  - Includes relationship:
    - *Mandatory* functionality of one or more use cases
  - Extends relationship
    - *Optiona*l functionality of a use case
  - Their semantics and notations

Use Cases Relationships: Overview

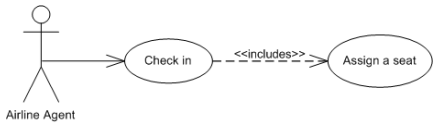
- Two types of relationships among use cases
  - A <<include>> B
  - C <<extend>> A



<<include>> Relationships



- **A** <<include>> **B**
  - **B** *is included in* **A**, and **A** *uses* **B**.
  - Use when:
    - B is a **mandatory functionality** of A; B is separated to simplify A
    - B is used by more than one base use case (The UML standard)
    - B has a volatile requirement that may change later
  - An *include* relationship is rendered as a *dependency*, *stereotyped* using ***guillemets*** as in <<include>>.
  - "«»" is pronounced as "Gee-**may".**



An Example Use Case Diagram with <<Includes>> relationship

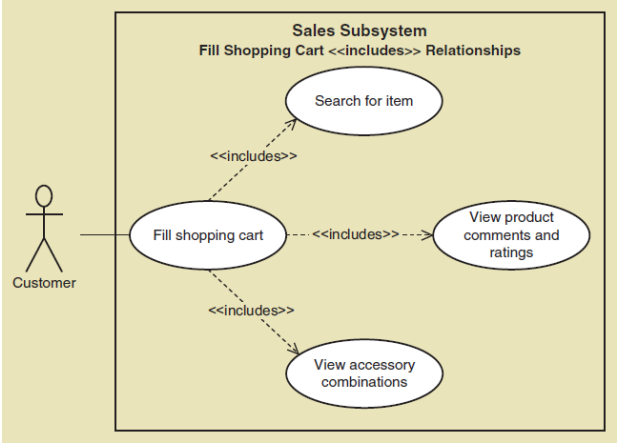
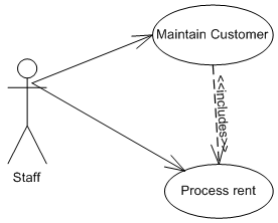


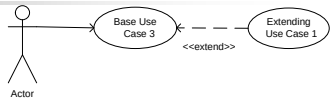
Figure 3-16

<<include>> Relationships

- An *included* use case could be another primary use case.



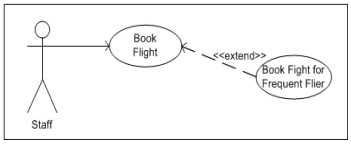
‹‹extend›› Relationships

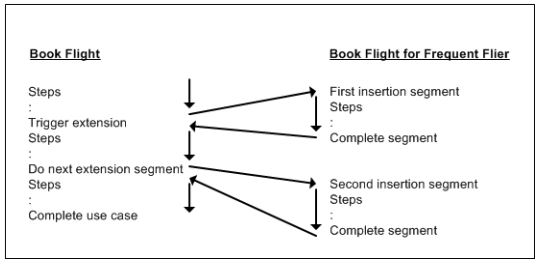


- B ‹‹extend›› A
  - **A** is a *base* use case; **B** is an *extending use case*.
  - **B** represents an ***optional* functionality** of the base use case **A**.
  - An *extend* relationship is rendered as a *dependency*, stereotyped as ‹‹extend››.

‹‹extend›› Relationships

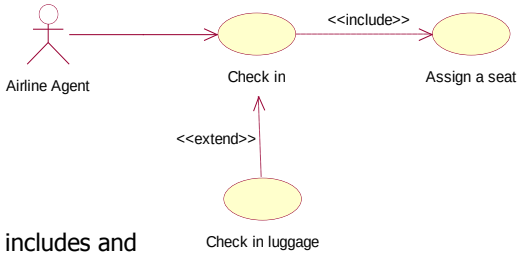
- B ‹‹extend›› A
  - **A** is a *base* use case; **B** is an *extending use case*.
  - **A** must declare certain "extension points," and **B** may add additional behavior only at those extension points.
  - An extension point should have a label (name) and a step number





Example: <<include>> & <<extend>> Relationships

- **Examples of *include* and *extend* use cases in the airline domain**
- Note the use of dependency (a dotted arrow) with the stereotype <<include>> and <<extend>>.
- Note the directions of arrows are different in <<include>> and <<extend>>.

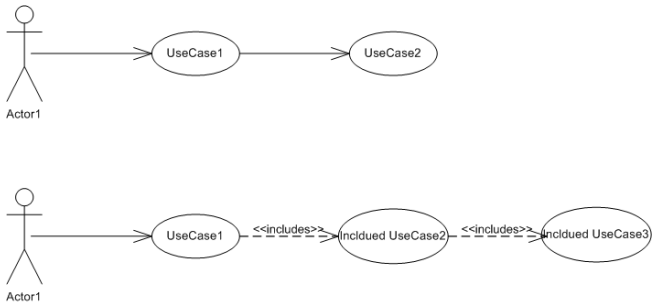


• Give another examples of includes and extends!

Recommendation: <<include>> & <<extend>>

- Use <<include>> use cases only when
  - It is a **mandatory functionality** AND
  - It is *worthy of a separate use case description*
- Warnings:
  - Do not overuse <<include>> use cases
    - *Apply template test:* An <<include>> use case must be complex enough to have a separate use case documentation
  - Do not nest <<include>> use cases
- Use <<extend>>
  - To add **optional** functionality

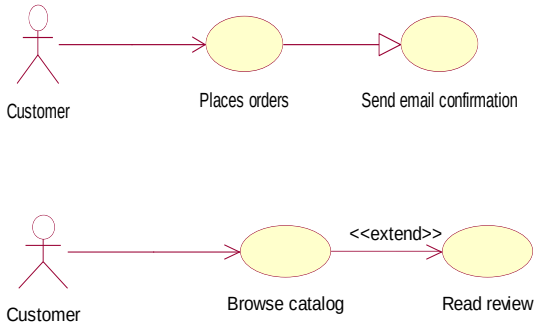
Patterns of bad use case diagrams



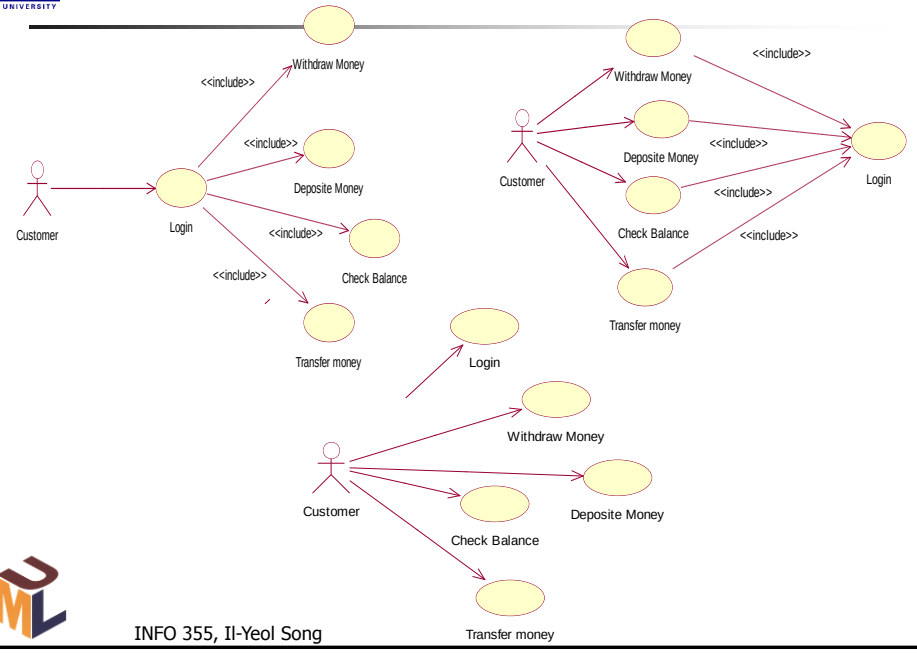
Guiding Principles

- The use case diagram is NOT a process flow
  - Avoid nested inclusion use cases
- Do not create too many small fragments of operations as inclusion use cases
  - Apply the Template test to each Included/extending use cases

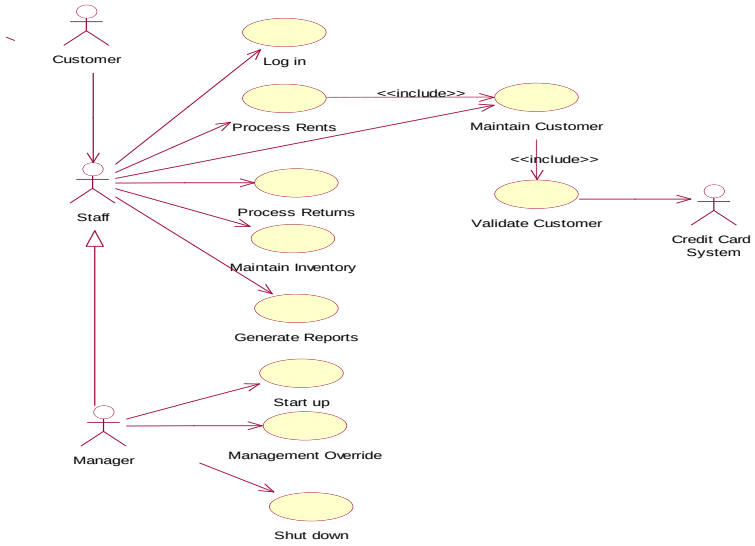
What's wrong in the following use cases in online bookstore domain?



Which is the best use case diagram?



Exercise: Identify some include/extend use cases in VRS



Demo for Using Visio for Use Case Diagrams

Review

- When do we use <<Include>> relationship?
- When do we use <<Extend>> relationships?
- How are their notations different?
- How do we call <<Include>> and <<Extend>> in the UML terminology?
- What is the name of the dotted arrow in the UML?

- What are two warnings in using <<Include>>? Why?