

Use Case Diagrams

Tutorial

What is a use case?

- A requirements analysis concept
- A case of a use of the system/product
- Describes the system's actions from a the point of view of a user
- *Tells a story*
 - A sequence of events involving
 - Interactions of a user with the system
- Specifies one aspect of the behavior of a system, *without specifying the structure of the system*
- Is oriented toward satisfying a user's goal

How do we describe use cases?

- Textual or tabular descriptions
- User stories
- Diagrams

Use Case Descriptions

- **actors** - something with a behavior or role, e.g., a person, another system, organization.
- **scenario** - a specific sequence of actions and interactions between actors and the system, a.k.a. a *use case instance*
- **use case** - a collection of related success and failure scenarios, describing actors using the system to support a goal.

What is an Actor?

- Include all user roles that interact with the system
- Include system components only if they responsible for initiating/triggering a use case.
 - For example, a timer that triggers sending of an e-mail reminder
- **primary** - a user whose goals are fulfilled by the system
 - importance: define user goals
- **supporting** - provides a service (e.g., info) to the system
 - importance: clarify external interfaces and protocols
- **offstage** - has an interest in the behavior but is not primary or supporting, e.g., government
 - importance: ensure all interests (even subtle) are identified and satisfied

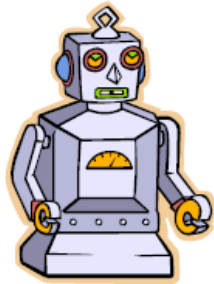
Finding Actors [1]

External objects that produce/consume data:

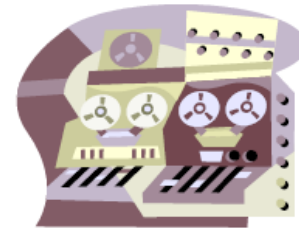
- Must serve as sources and destinations for data
- Must be external to the system



Humans



Machines



External systems



Organizational Units



Sensors

Finding Actors [2]

Ask the following questions:

- Who are the system's **primary users**?
- Who **requires system** support for **daily tasks**?
- Who are the system's **secondary users**?
- What **hardware** does the system handle?
- Which **other (if any) systems** **interact** with the system in question?
- Do any entities interacting with the system perform multiple roles as actors?
- Which other **entities** (human or otherwise) might have an interest in the system's output?

What is a user story?

- An abbreviated description of a use case
- Used in [agile development](#)

Answers 3 questions:

- 1.Who?
- 2.Does what?
- 3.And why?

*As a <type of user>,
I want <some behavior from the system>
so that <some value is achieved>*



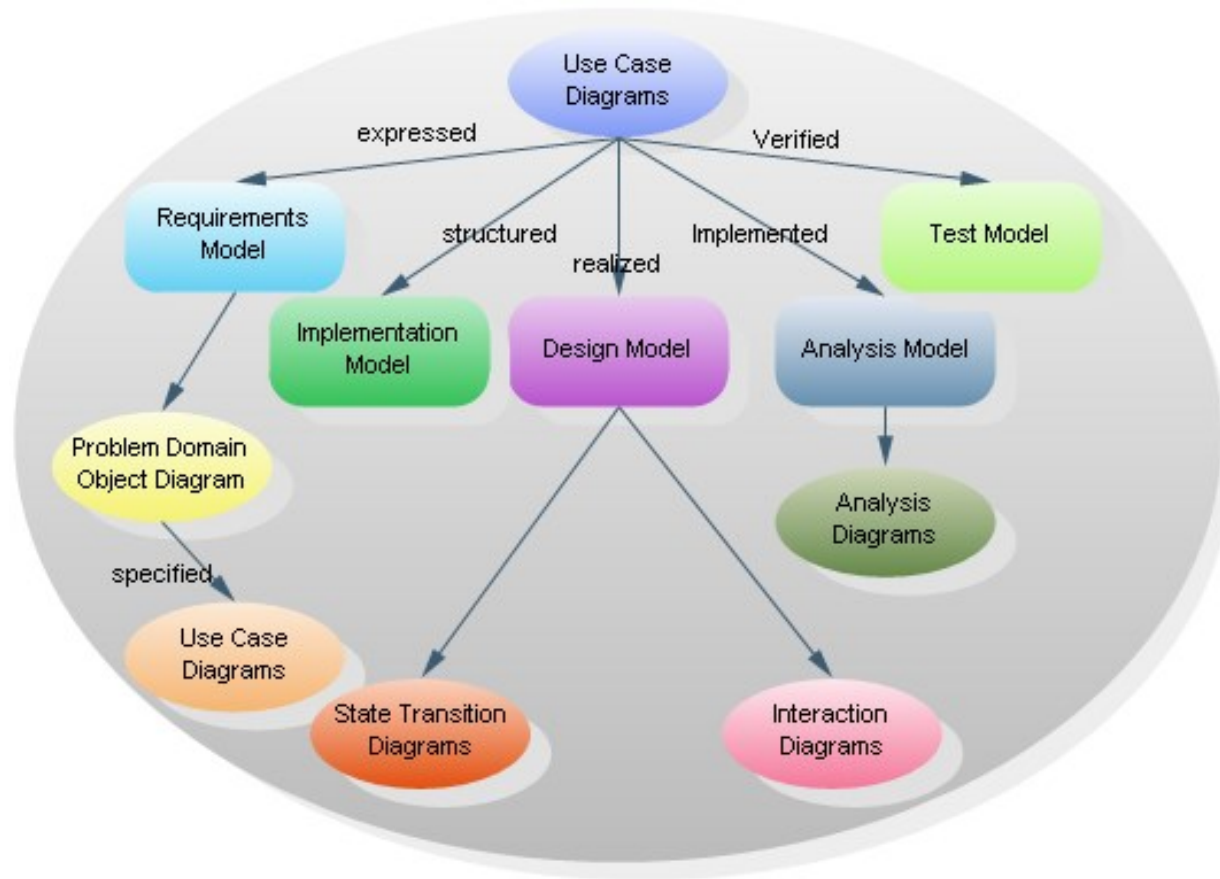
Use Case Diagrams

- A picture
 - describes how actors relate to use cases
 - and use cases relate to one another
- Diagrams are not essential
- They are helpful in giving an overview, but only secondary in importance to the textual description
- They do not capture the full information of the actual use cases
- In contrast, text *is* essential

Use Case Diagram **Objective**

- Built in early stages of development
- Purpose
 - Specify the context of a system
 - Capture the requirements of a system
 - Validate a systems architecture
 - Drive implementation and generate test cases
 - Developed by analysts and domain experts

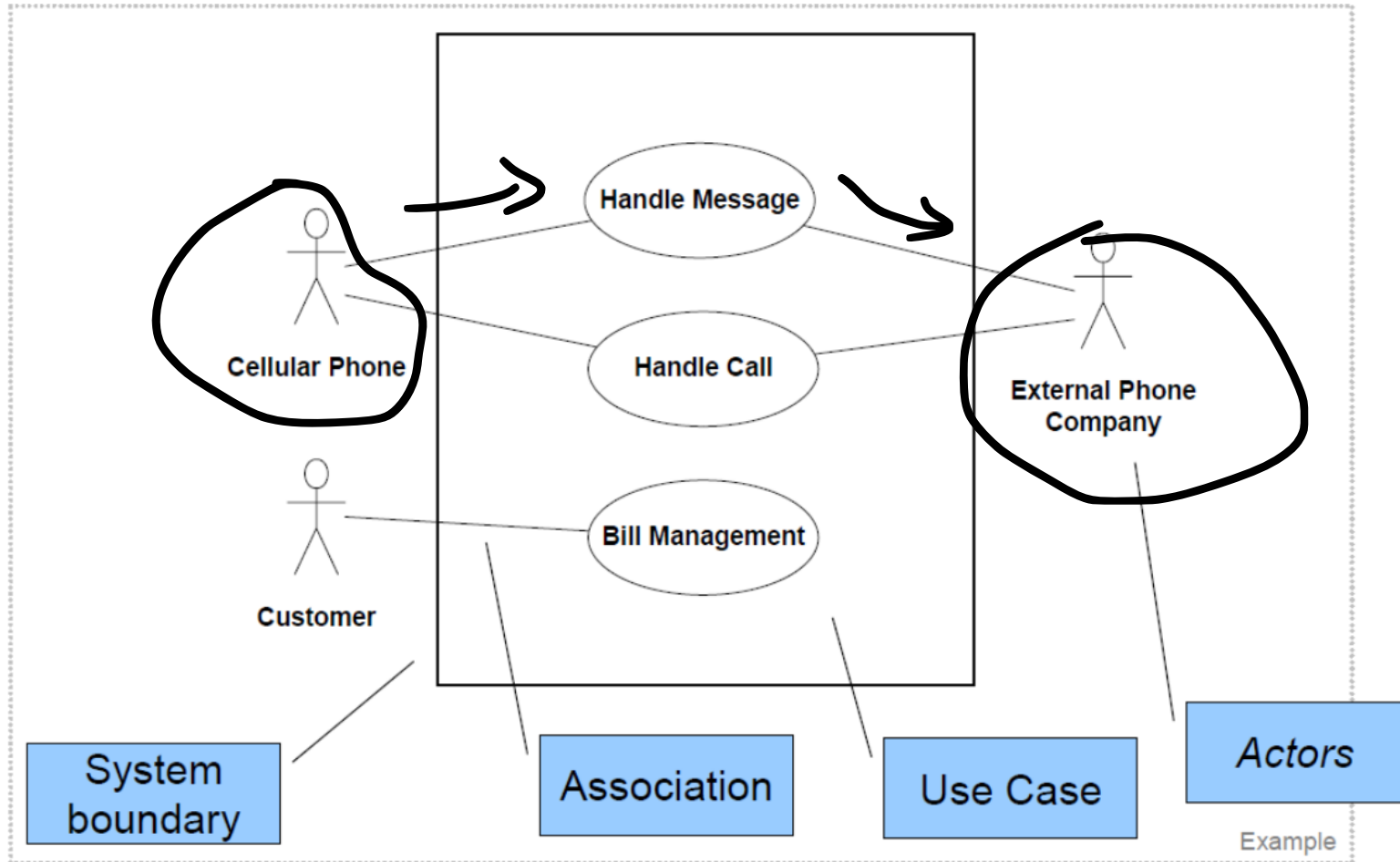
How do use case diagrams fit in?



This applies also to use case descriptions.

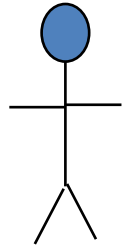
Diagram reproduced from www.edrawsoft.com.

Example Use-Case Diagram



A standard form of use case diagram is defined in the Unified Modeling Language.

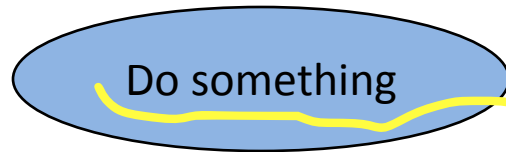
Elements of use case diagram: Actor



Name

- *Actor* is someone interacting with use case (system function). Named by noun.
- Similar to the concept of user, but a user can play different *roles*; (example: a prof. can be instructor and researcher – plays 2 roles with two systems).
- Actor *triggers* use case.
- Actor has responsibility toward the system (inputs), and Actor have expectations from the system (outputs).

Elements of use case diagram: Use Case



- System function (process – automated or manual).
- Named by verb.
- Each Actor must be linked to a use case, while some use cases may not be linked to actors.

USER/ACTOR	USER GOAL = Use Case
Order clerk	Look up item availability Create new order Update order
Shipping clerk	Record order fulfillment Record back order
Merchandising manager	Create special promotion Produce catalog activity report

Elements of use case diagram: Other details

———— Connection between Actor and Use Case



Boundary of system

<<include>>

Include relationship between Use Cases (one UC must call another; e.g., Login UC includes User Authentication UC)

<<include>>

<<extend>>

Extend relationship between Use Cases (one UC calls Another under certain condition; think of if-then decision points)

Linking Use Cases

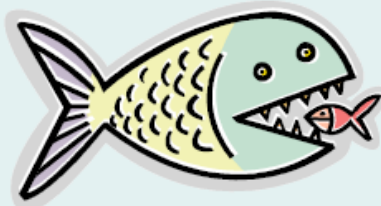
- *Association* relationships
- *Generalization* relationships *base*
 - One element (child) "is based on" another element (parent)
- *Include* relationships
 - One use case (base) includes the functionality of another (inclusion case)
 - Supports re-use of functionality
- *Extend* relationships
 - One use case (extension) extends the behavior of another (base)

Use Case Levels

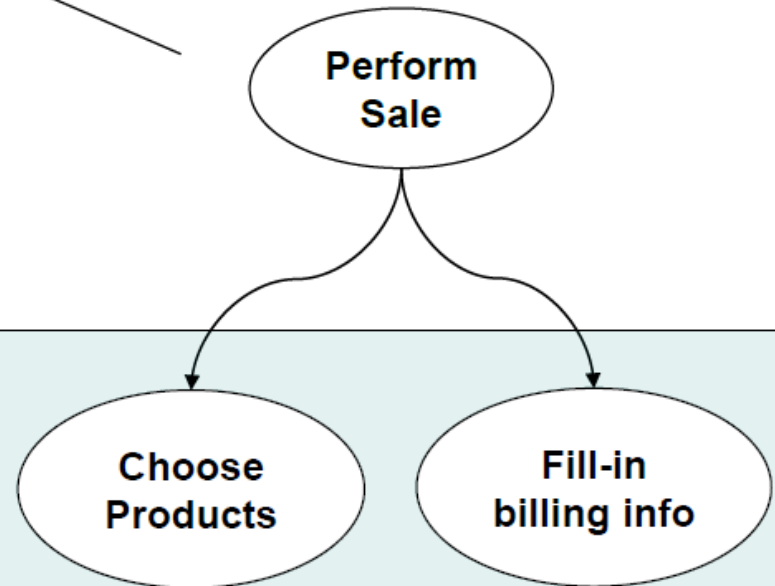
Base Use Case:
Used directly by
the user

↑ User goals

↓ Sub-functionality

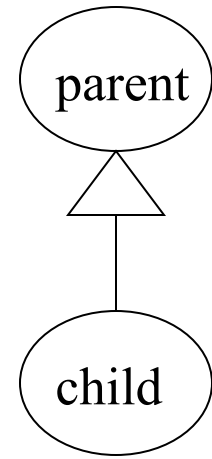


Alistair Cockburn "Writing Effective Use Cases"

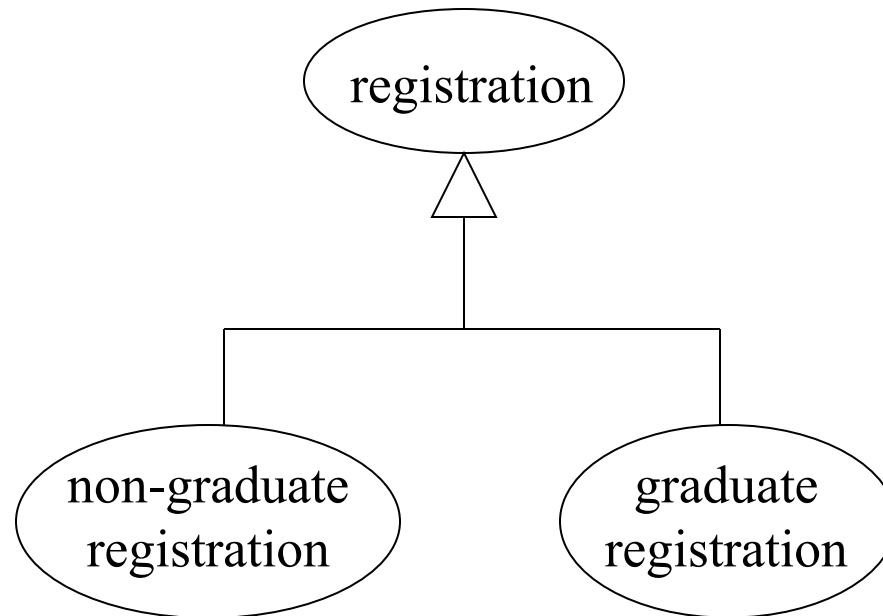


1. Generalization

- The child use case inherits the behavior and meaning of the parent use case.
- The child may add to or override the behavior of its parent.



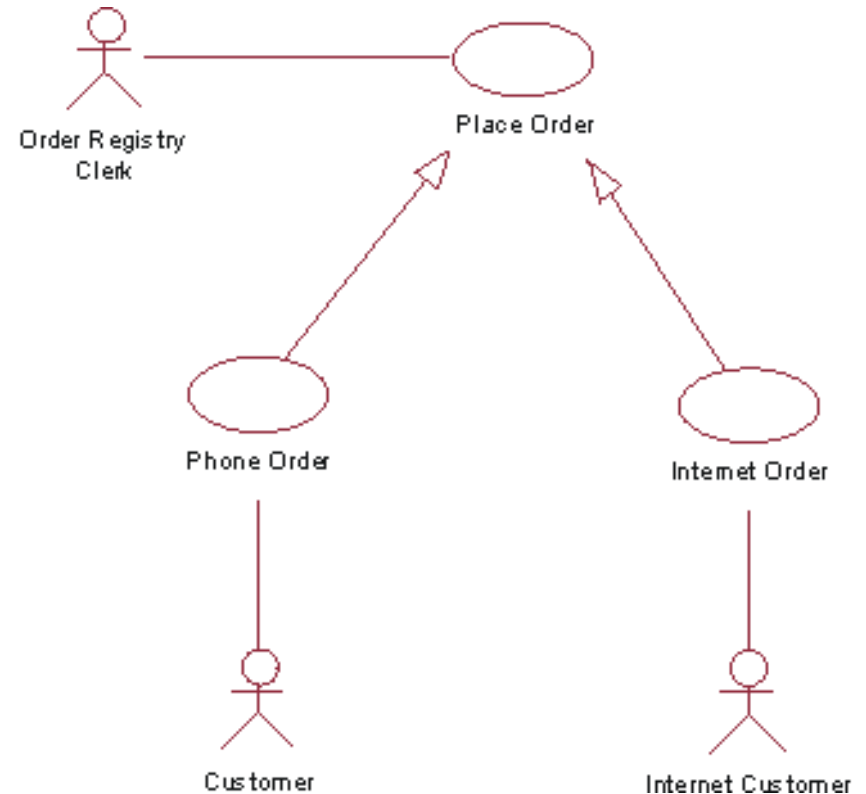
More about Generalization



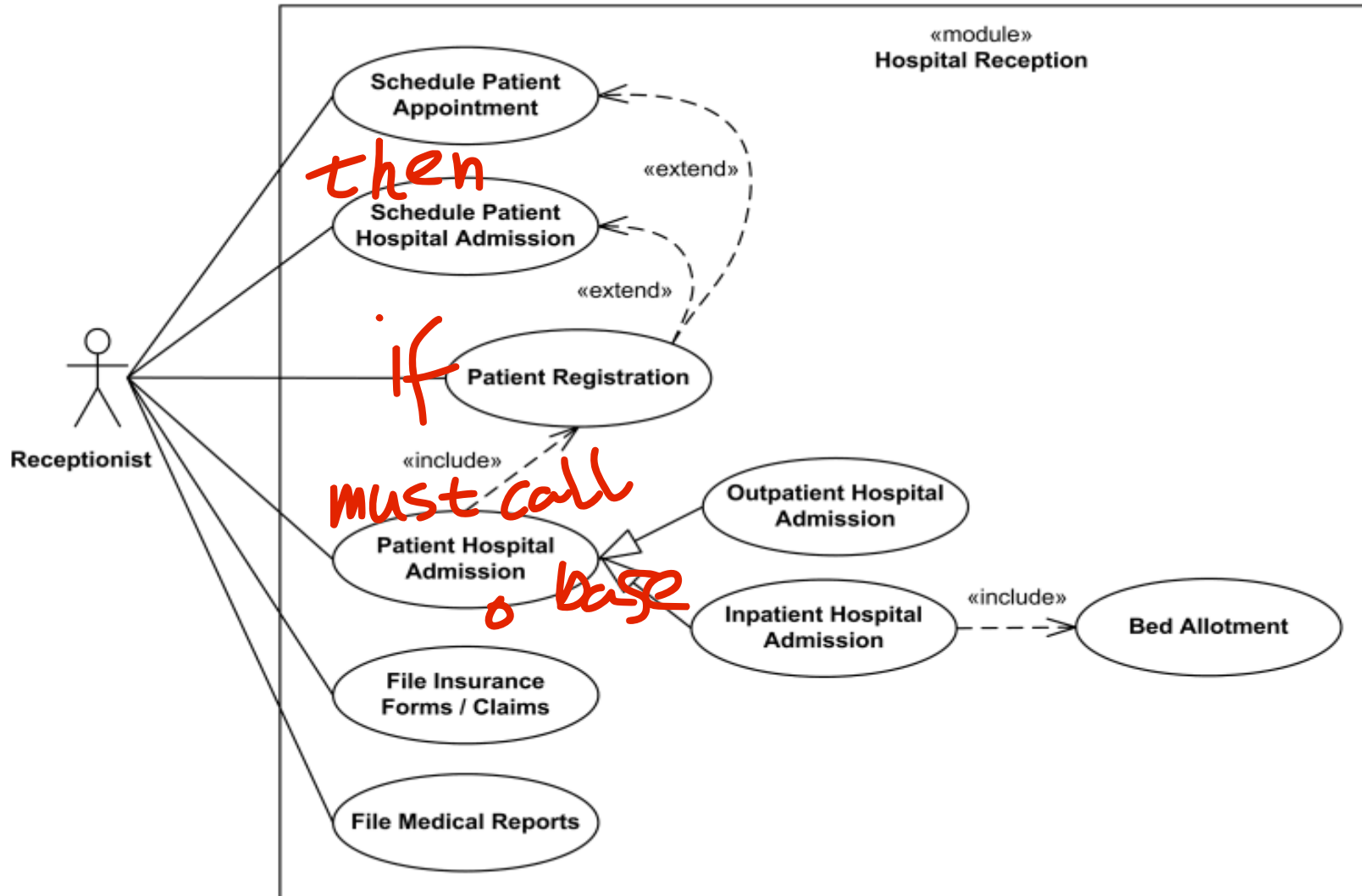
Generalization Example

The actor Order Registry Clerk can instantiate the general use case Place Order.

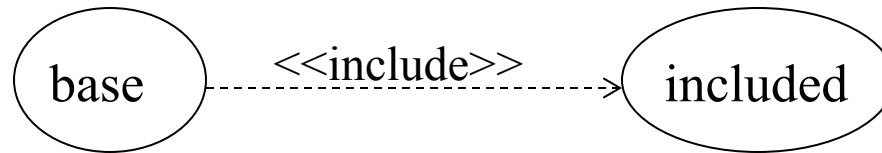
Place Order can also be specialized by the use cases Phone Order or Internet Order.



Generalization Example



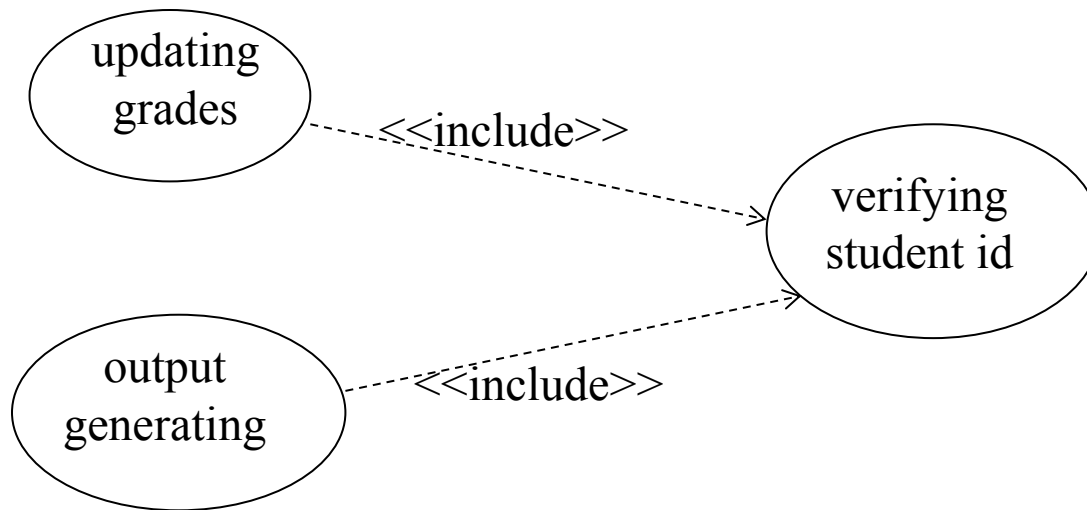
2. Include



- The base use case explicitly ^{must} incorporates the behavior of another use case at a location specified in the base.
- The included use case never stands alone. It only occurs as a part of some larger base that includes it.

More about Include

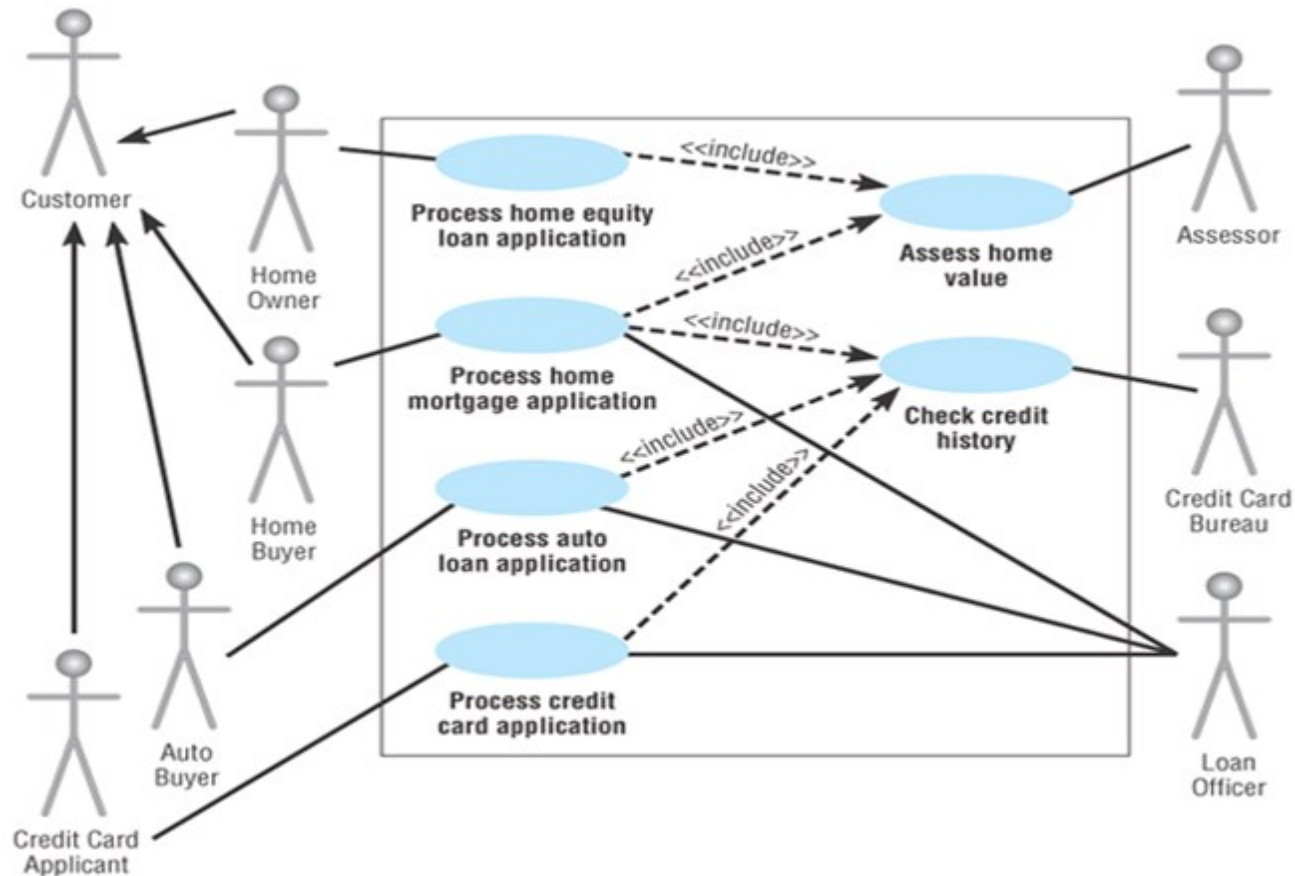
Enables us to avoid describing the same flow of events several times by putting the common behavior in a use case of its own.



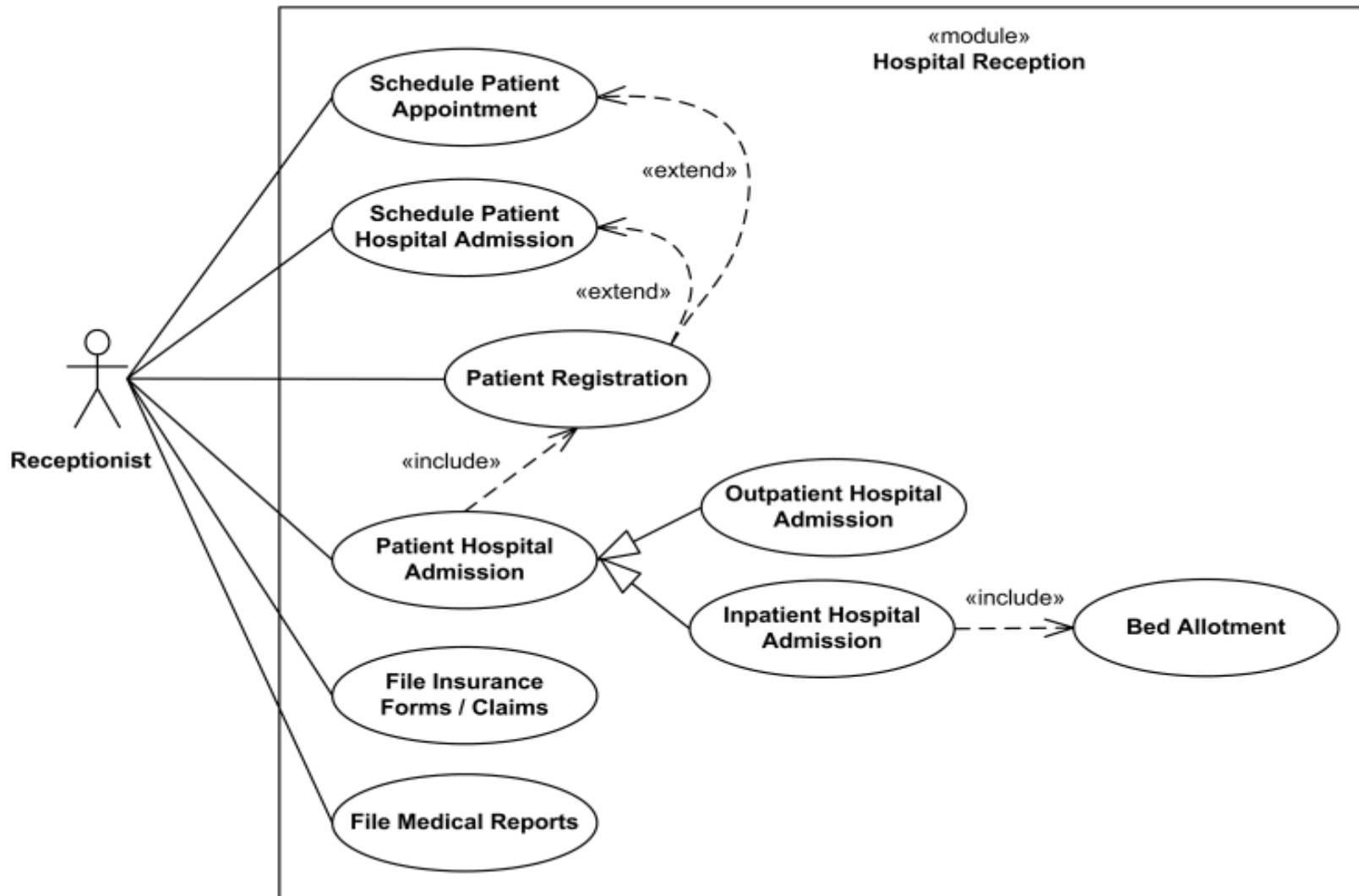
Include relationship

- Include relationship – a standard case linked to a **mandatory** use case.
- Example: to *Authorize Car Loan* (standard use case),
a clerk must run *Check Client's Credit History* (include use case).
- The standard UC includes the mandatory UC (use the verb to figure direction arrow).
- Standard use case can NOT execute without the include case → **tight coupling** .

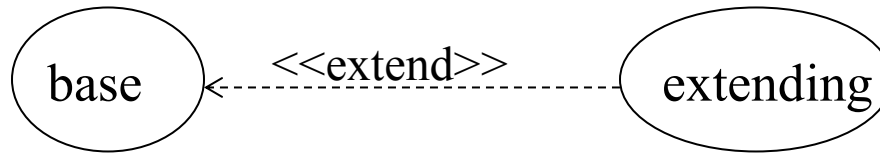
Reading use case diagram with **Include** relationship



Include Example



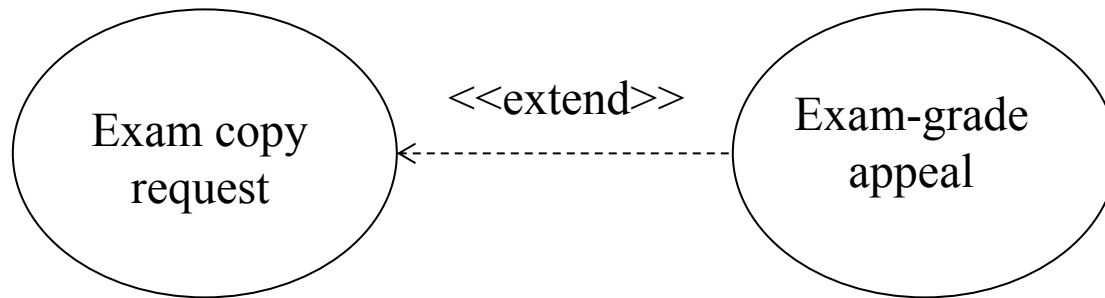
3. Extend



- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.

More about **Extend**

- Enables to model optional behavior or branching under conditions.

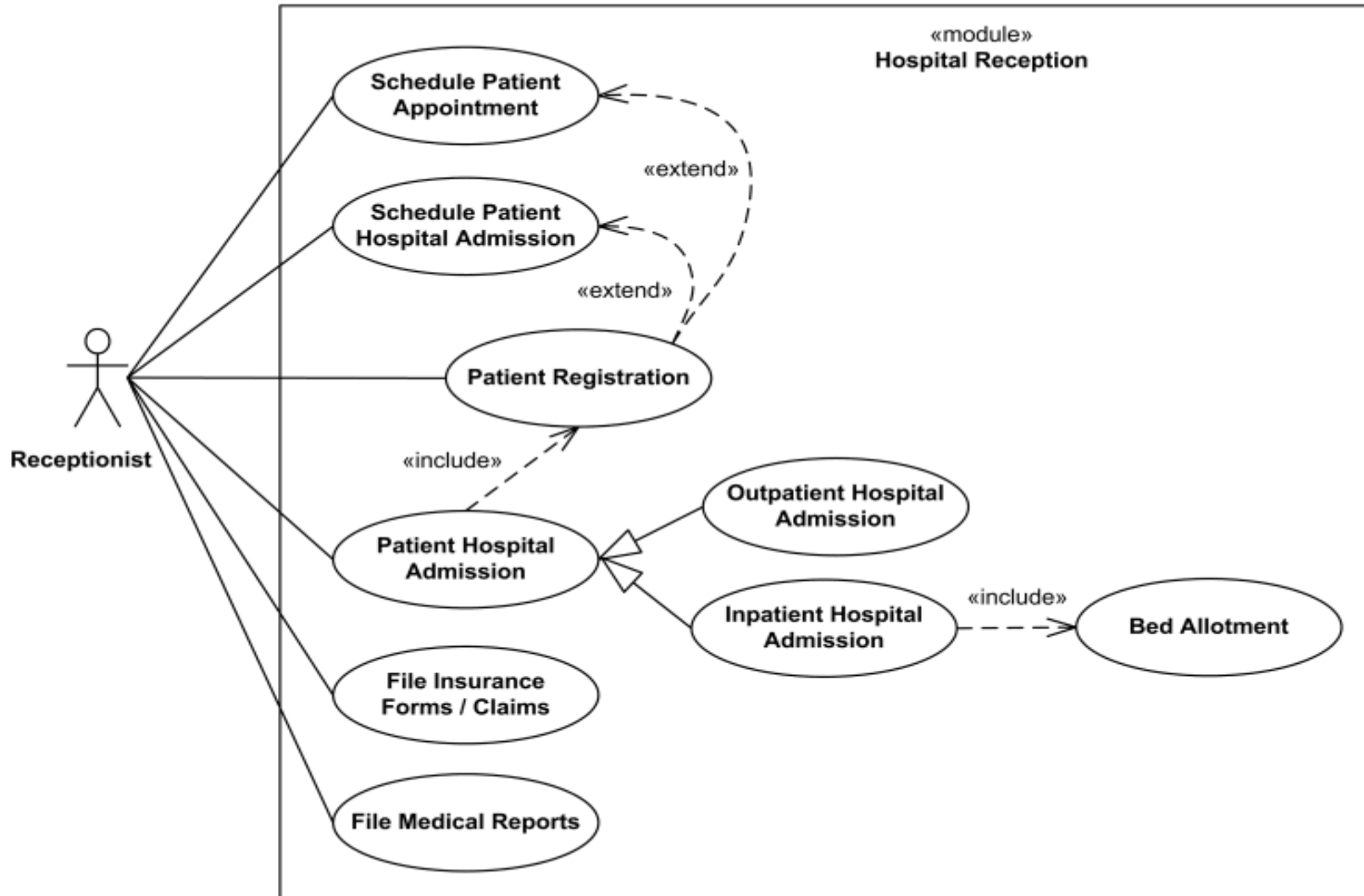


Extend relationship

- Extend relationship – linking an **optional** use case to a standard use case.
- Example: *Register Course* (standard use case) may have *Register for Special Class* (extend use case) – class for non-standard students, in unusual time, with special topics, requiring extra fees...).
- The optional UC extends s the standard UC
- Standard use case can execute without the extend case
→ loose coupling.

[Reading extend relationship](#)

Extend Example #1



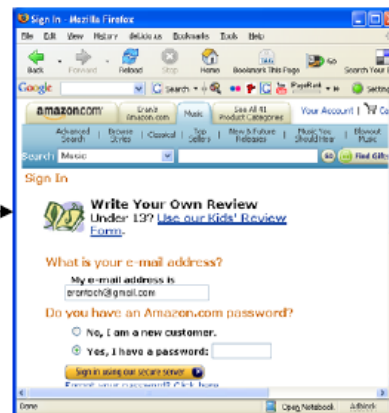
Extend Example #2



Product Page

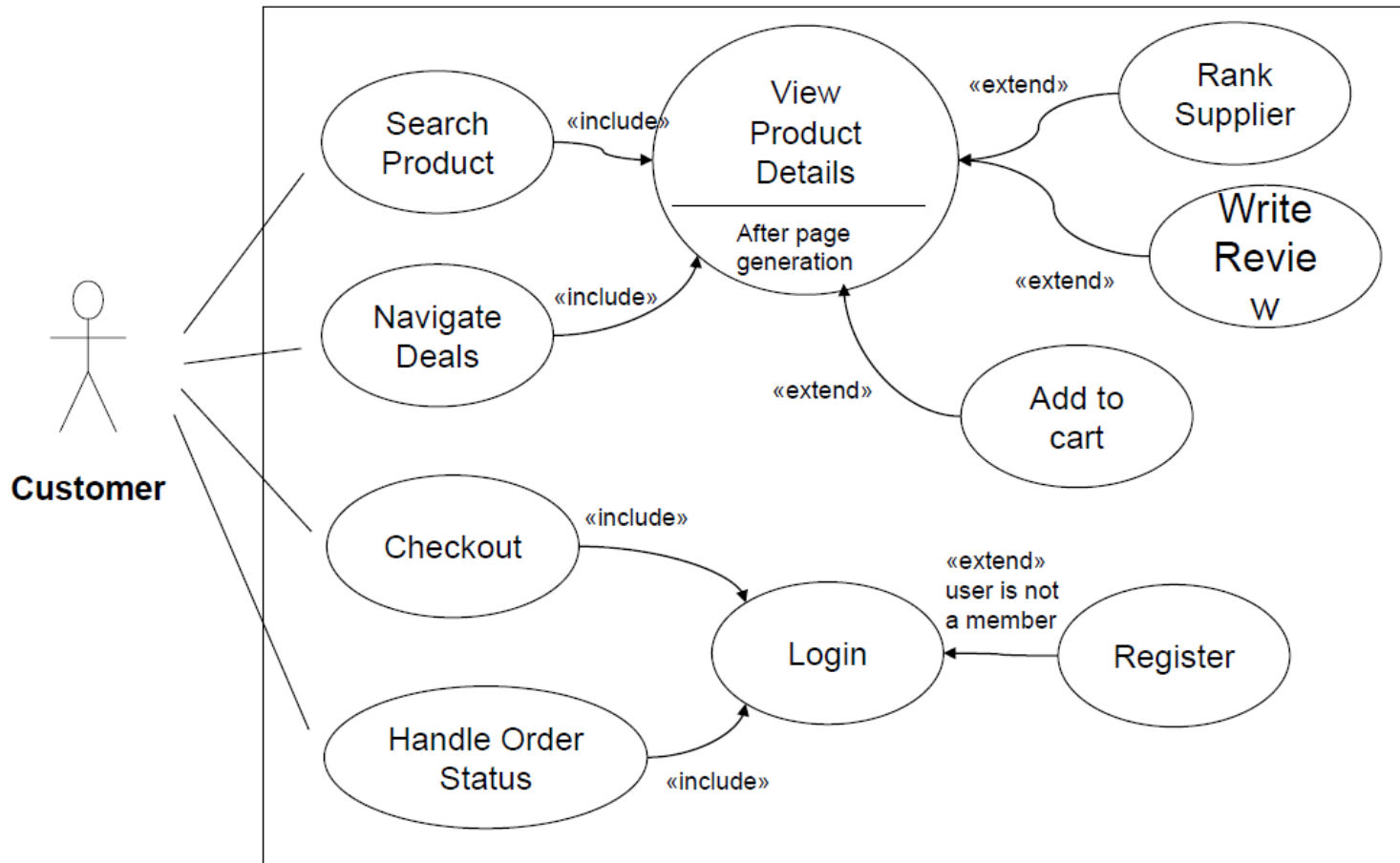


Shopping Cart



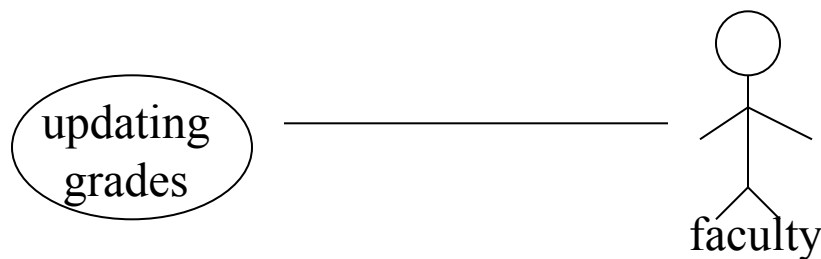
Review Writing

Extend Example #2 cont.

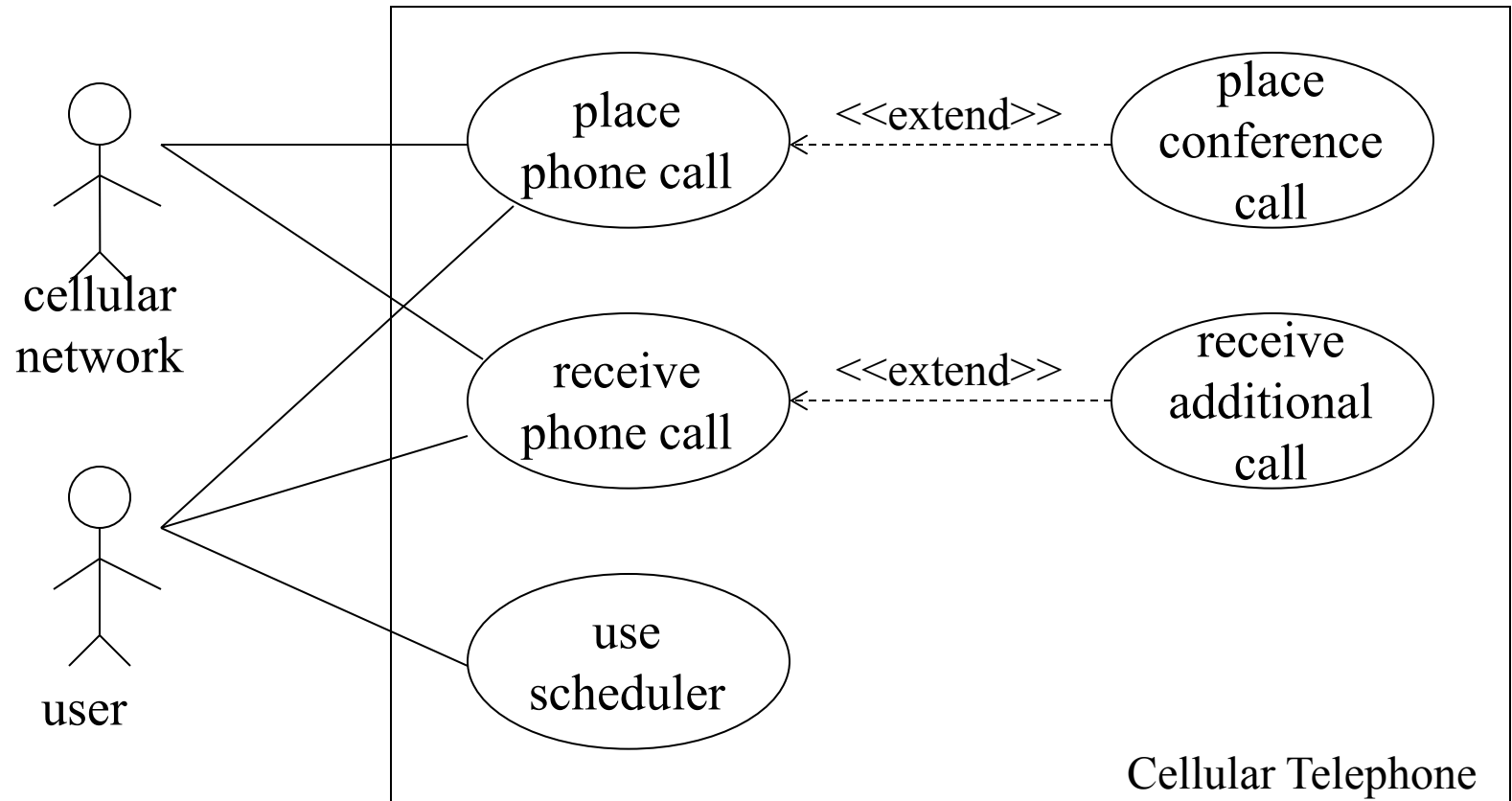


Relationships between Use Cases and Actors

Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.



Example #1



Example #2

Altered State University (ASU) Registration System

1. Professors indicate which courses they will teach on-line.
2. A course catalog can be printed
3. Allow students to select on-line four courses for upcoming semester.
4. No course may have more than 10 students or less than 3 students.
5. When the registration is completed, the system sends information to the billing system.
6. Professors can obtain course rosters on-line.
7. Students can add or drop classes on-line.

Example #2 cont.

Altered State University (ASU) Registration System



How to create use case diagram

1. List main system functions (use cases) in a column:

- think of business events demanding system's response
- users' goals needs to be accomplished via the system
- Create, Read, Update, Delete (CRUD) data tasks
- Naming use cases – user's needs usually can be translated in data tasks

2. Draw ovals around the function labels

3. Draw system boundary

4. Draw actors and connect them with use cases (if more intuitive, this can be done as step 2)

5. Specify include and extend relationships between use cases (yes, at the end - not before, as this may pull you into process thinking, which does not apply in UC diagramming).

Use-Case Diagrams: Example [1]

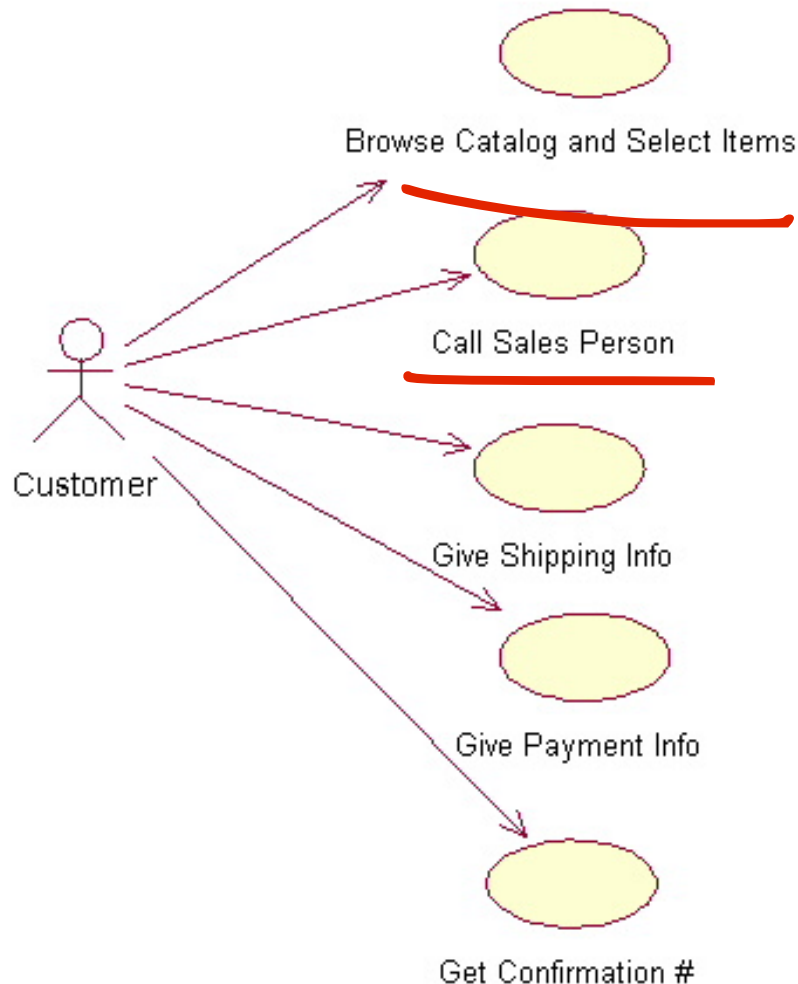
I. Begin with a Use Case!

A user placing an order with a sales company might follow these steps :

1. Browse catalog and select items.
2. Call sales representative.
3. Supply shipping information.
4. Supply payment information.
5. Receive conformation number from salesperson.

II. Then translate Use Case sequence into Diagram

Use-Case Diagrams: Example [2]



The salesperson could also be included in this use case diagram because the salesperson is also interacting with the ordering system.

Use-Case Diagram Case Study [1]

Vending Machine

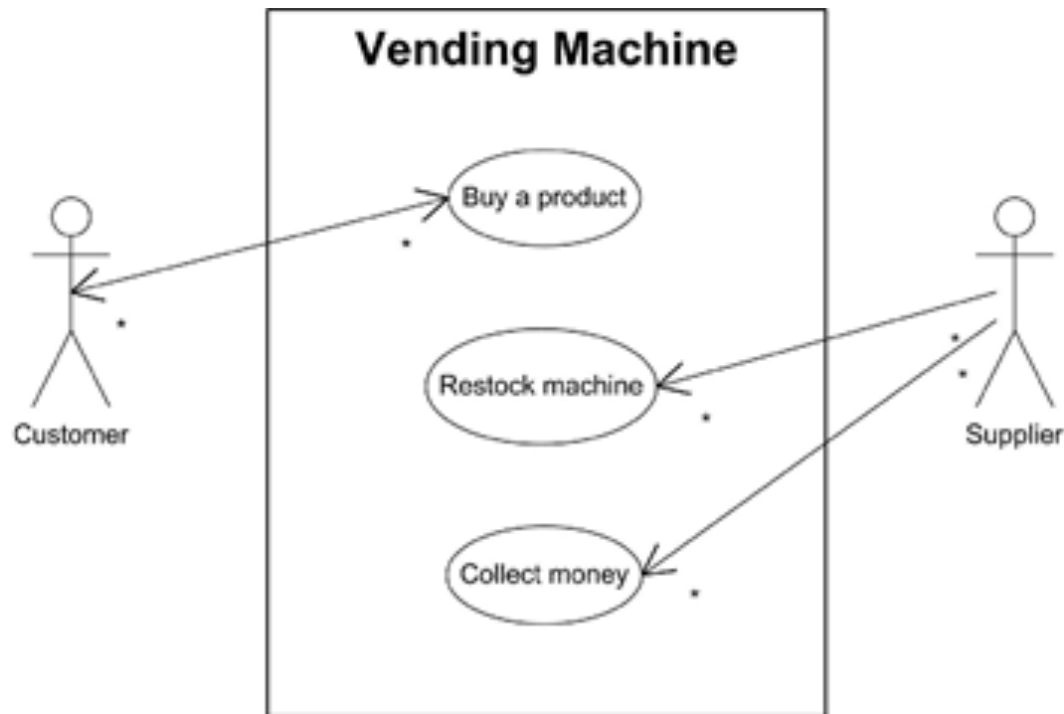
After client interview the following system scenarios were identified:

- A customer buys a product
- The supplier restocks the machine
- The supplier collects money from the machine

On the basis of these scenarios, the following three actors can be identified:

Customer; Supplier; Collector (in this case Collector=Supplier)

Use-Case Diagram Case Study [2]



Use-Case Diagram Case Study [3]

Introducing annotations (notes) and constraints.

