Writing Use Cases

Lab Lecture 4

2015/04/08

MINING MINDS

UCL Ubiquitous Computing Laboratory

Kyung Hee University, Korea

What is a Use case?

- written descriptions of user's interaction with the software product to accomplish a goal
  - (in a business system): "A sequence of transactions in a system whose task is to yield a result of measurable value to an individual actor of the business system”
  - (in an information system): "A behaviorally related sequence of transactions performed by an actor in a dialogue with the system to provide some measurable value to the actor" (Jacobson 1995)
- Use cases are the ways in which a system can be used (the functions which the system provides to its users)
- Use cases help us discover/document requirements

Benefits of Use cases

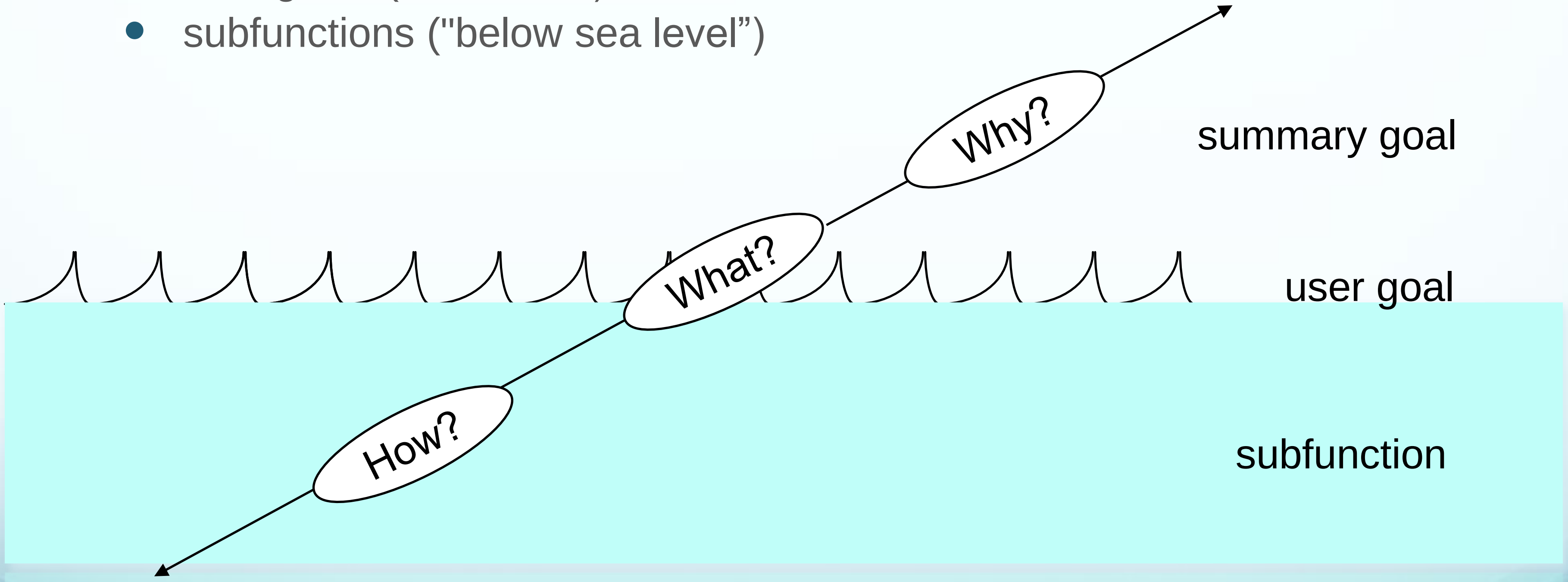
- The list of goal names provides executives:
  - Shortest summary of what system will contribute
  - Project planning skeleton (priorities & timing)
- The main success scenario provides all:
  - Agreement as to the system's responsibilities
- The extension conditions provide programmers:
  - List of things programmers have to watch for
  - List of things analysts have to investigate
- The extension handling steps provide dev team:
  - Record of (tricky) business policy decisions

Actors and Stakeholders

- Actor:
  - actor: anything with behavior that acts on the system
  - primary actor: initiates interaction to achieve goal
    (when system is a software product, primary actor is often the computer user)
  - supporting actor: performs sub-goals to help use case
- Actor vs. Stakeholder:
  - stakeholder: anyone interested in the system
  - examples: supplier, stock agency, vendor
  - the difference: stakeholder might not "act" in any scenario

Use case goals and levels

- goal: action that actor wants to accomplish
- level: type / scope of a goal
  - summary goals ("above sea level”)
  - user goals ("sea-level”)
  - subfunctions ("below sea level”)



Use case goals and levels

- Withdraw money from an ATM
  - level?
    - User goal
- Purchase a book from the online store, and have it shipped to the user; can be cancelled while in transit
  - level?
    - Summary goal
- Purchase shares of stock online using a "stock trap."
  - level?
    - Summary goal
- Update user's balance after a deposit.
  - level?
    - subfunction

Qualities of a good use case

- a good use case:
  - starts with a request from an actor to the system
  - ends with the production of all the answers to the request
  - defines the interactions (between system and actors) related to the function
  - takes into account the actor's point of view, not the system's
  - focuses on interaction, not internal system activities
  - doesn't describe the GUI in detail
  - has 3-9 steps in the main success scenario
  - is easy to read

Use case vs. Internal Features

- consider software to run a cell phone:

Use Cases

- call someone
- receive a call
- send a message
- memorize a number

Point of view: user

Internal Functions

- transmit / receive data
- energy (battery)
- user I/O (display, keys, ...)
- phone-book mgmt.

Point of view: developer / designer

How about these Requirements ?

- Which of these requirements should be represented directly in a use case?
  - Order cost = order item costs * 1.06 tax.
  - Promotions may not run longer than 6 months.
  - Customers only become Preferred after 1 year.
  - A customer has one and only one sales contact.
  - Response time is less than 2 seconds.
  - Uptime requirement is 99.8%.
  - Number of simultaneous users will be 200 max.

How about these Requirements ?

- Which of these requirements should be represented directly in a use case?
  - Order cost = order item costs * 1.06 tax.
  - Promotions may not run longer than 6 months.
  - Customers only become Preferred after 1 year.
  - A customer has one and only one sales contact.
  - Response time is less than 2 seconds.
  - Uptime requirement is 99.8%.
  - Number of simultaneous users will be 200 max.

Answer: ***NONE***. Most of these requirements are non-functional, so the use cases wouldn't explicitly mention them. The user doesn't see them directly in the success scenario.

Styles of Use cases

1. actor / goal list or UML use case diagram
   - shows all use cases in system
2. informal use case
3. formal use case

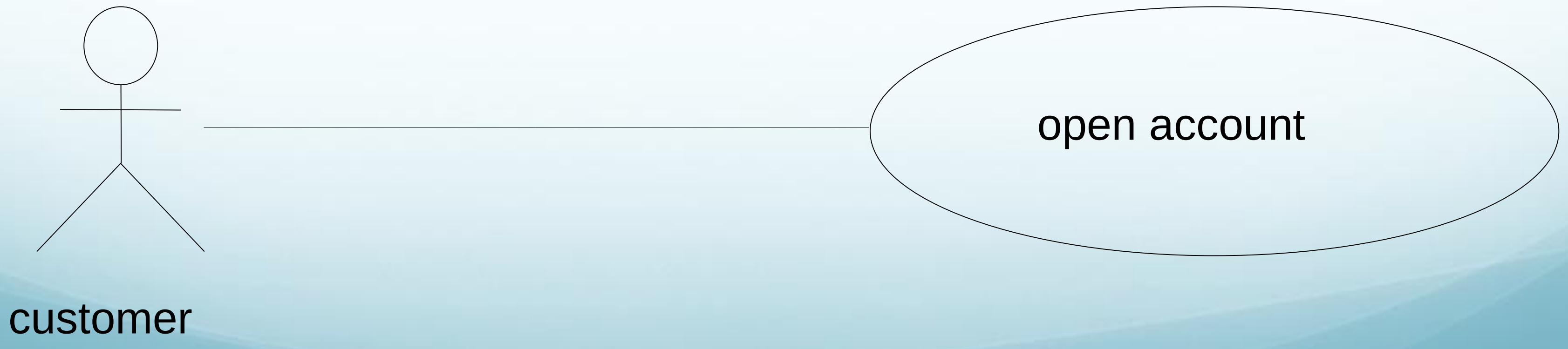
1. Actor / goal list

- it can be useful to create a list or table of actors and their "goals" (use cases they start):

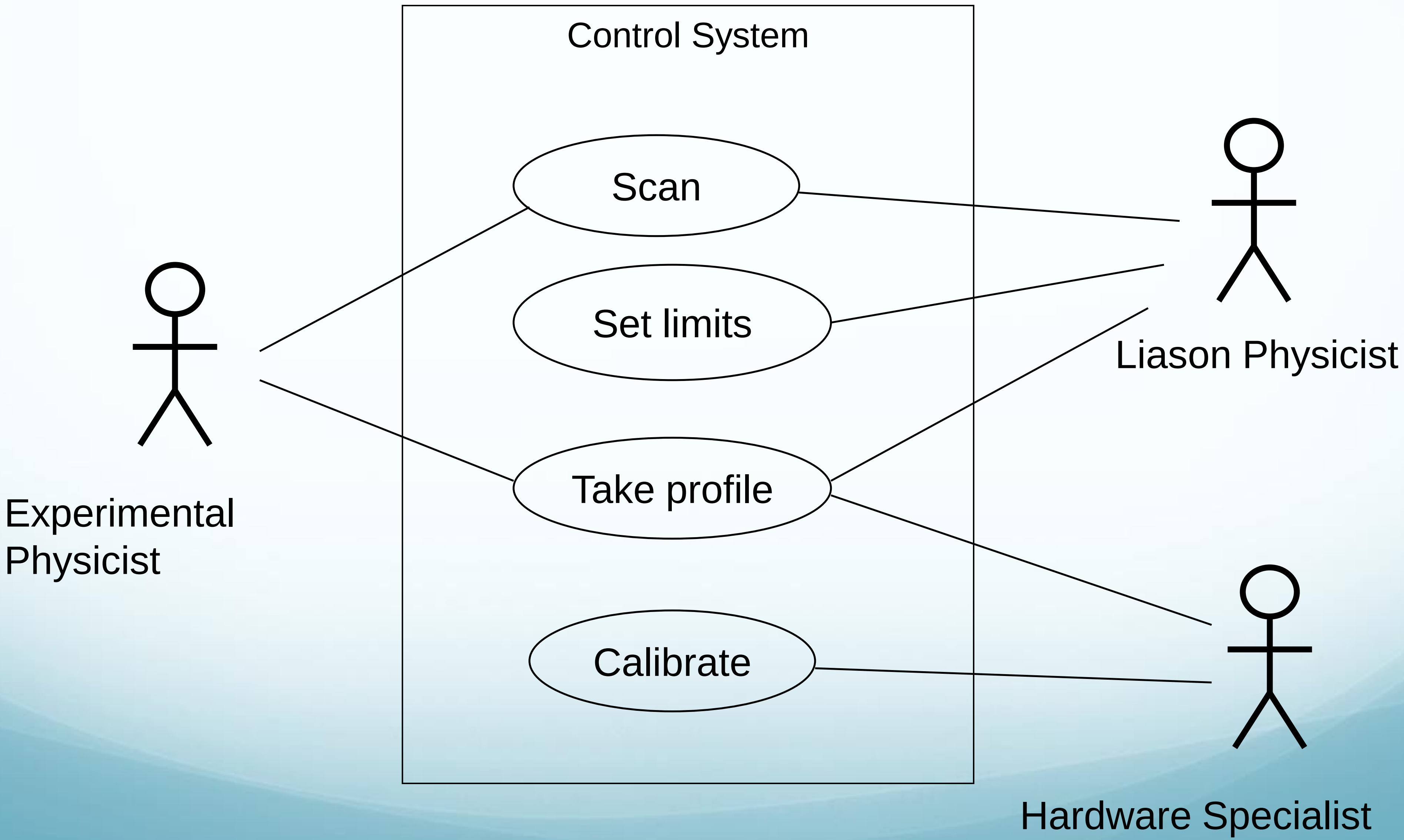
ACTORS		USE CASE
Club Member	Initiates	Submit Promotion Order Submit Regular Order
Potential Member	Initiates	Submit New Subscription
Past Member	Initiates	Submit Subscription Renewal
Membership Services Dept.	Initiates	Request Membership
Marketing Department	Initiates	Create New Monthly Promotion Create New Seasonal Promotion Create New Subscription Program Request Promotion Reports Request Sales Reports
Membership Services System	Initiates	Send New Subscription Offer Send Club Promotion Send Subscription Renewal

1. UML Use case diagram

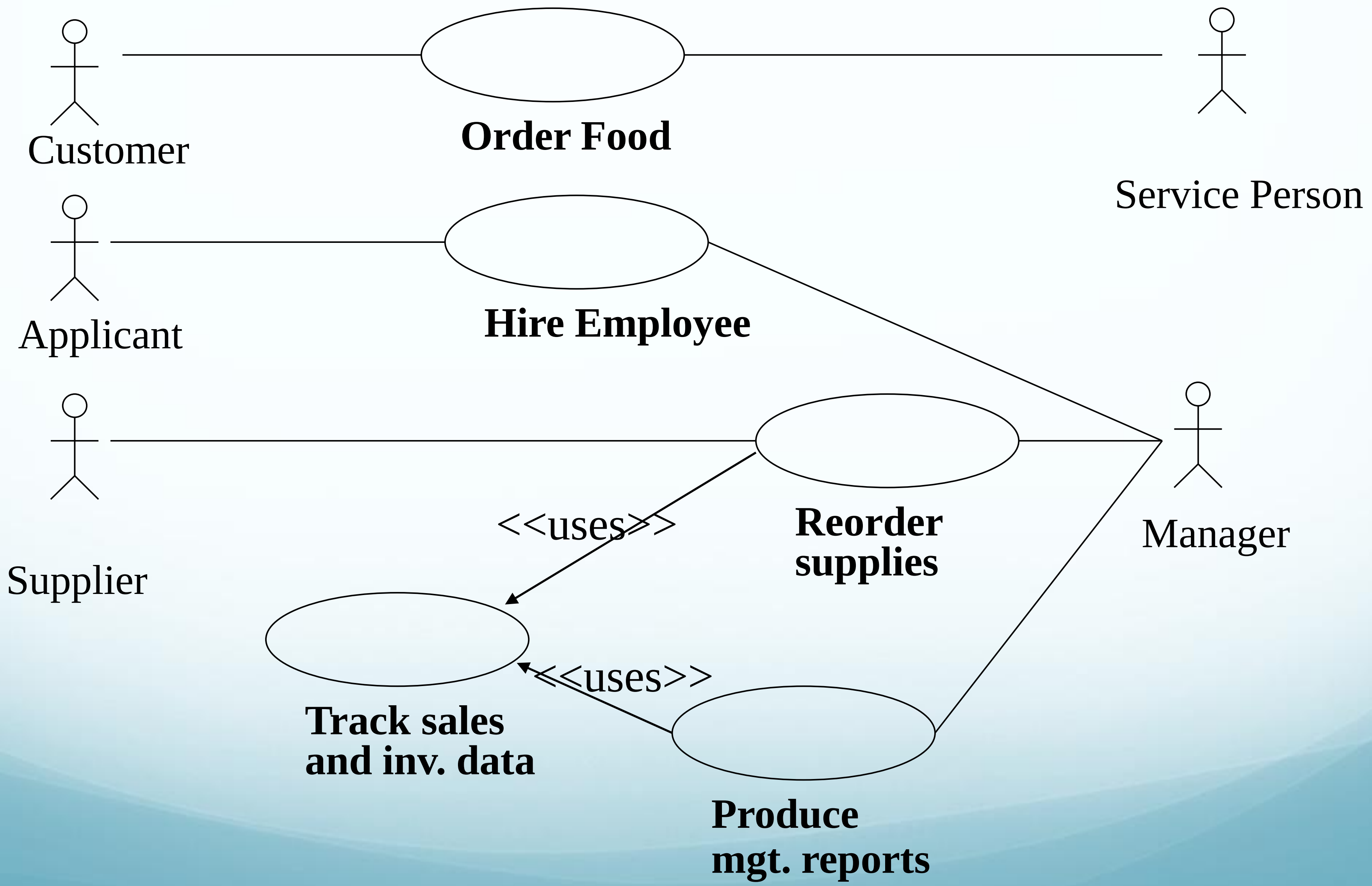
- use cases can be drawn as diagrams, with:
  - actors as stick-men, with their names below
  - use cases as ellipses with their names below or inside
  - association indicated by lines, connecting an actor to a use case in which that actor participates
  - use cases can be connected to other cases that they use / rely on



1. UML Use case diagram



1. UML Use case diagram



3. Formal use case

(goal of primary actor)

(level of goal [summary, user, subfunction])

(primary actor)

"Place an order" (User goal / Clerk)

Main scenario:

1. Clerk identifies customer, item and quantity.
2. System accepts and queues the order.

(action steps:
full sentences showing
who takes the action!
3 - 9 steps long.)

(condition causing different actions)

Extensions:

1a. Low credit & Customer is 'Preferred':
 1a1. System gives them credit anyway.

(action step(s)
handling those conditions)

1b. Low credit & not 'Preferred' customer:
 1b1. Clerk performs Sign Up Preferred Customer scenario and accepts only prepayment.

(calling another use case)

2a. Low on stock: Customer accepts rain-check:
 2a1. Clerk reduces order to available stock level.

3. Formal Use Case

USE CASE NAME	Submit Promotion Order
ACTOR	Club Member
DESCRIPTION	Describes the process when a club member submits a club promotion order to either indicate the products they are interested in ordering or declining to order during this promotion
Normal Course	1. This use is initiated when the club member submits the promotion order to be proceeded
	2. The club member's personal information such as address is validated against what is currently recorded in member services
	3. The promotion order is verified to see if product is being ordered
	4. The club member's credit status is checked with Accounts Receivable to make sure no payments are outstanding
	5. For each product being ordered, validate the product number
	6. For each product being ordered, check the availability in inventory and record the ordered information which includes "quantity being ordered" and give each ordered product a status of "open"
	7. Create a Picking Ticket for the promotion order containing all ordered products which have a status "open"
	8. Route the picking ticket to the warehouse
PRECONDITION	Use case *send club promotion* has been processed
POST CONDITION	Promotion order has been recorded and the picking ticket has been routed to the warehouse
ASSUMPTIONS	

Method to write use cases

1. identify actors and their goals
2. write the main success scenario
3. identify and list possible failure extensions
4. describe how the system handles each failure

1. Identify actors and goals

Ask oneself the following questions:

- what computers, subsystems and people will drive our system? (actors)
  - *examples: Customer, Clerk, Corporate Mainframe*
- what does each actor need our system to do?
  - each need may show up as a trigger to a use case

result: a list of use cases, a sketch of the system

- short, fairly complete list of usable system function
- can now draw UML use case diagram for reference

2. Write the main success scenario

- main success scenario is the preferred "happy" case
  - example: customer=good credit and item=in stock
  - easiest to read and understand
  - everything else is a complication on this

- capture each actor's intent and responsibility, from trigger to goal delivery
  - say what information passes between them
  - number each line

3. List the failure extensions

- usually, almost every step can fail
  - *example: customer has bad credit*
  - *example: item is not in stock in desired quantity*

- note the failure condition separately, after the main success scenario

4. Describe the failure handling

- recoverable extensions rejoin main course
  - *example: low credit + valued customer -> accept*
  - *example: low stock + reduce quantity -> accept*
- non-recoverable extensions fail directly
  - not a valued customer -> decline order
  - out of stock -> decline order
- each scenario goes from trigger to completion
  - "extensions" are merely a writing shorthand
  - can write "if" statements
  - can write each scenario from beginning to end

Pros and Cons of Writing Use cases

pro:

- they hold functional requirements in an easy-to-read text format
- they make a good framework for
  non-functional requirements & project scheduling

con:

- they show only the functional reqs
- design is not done only in use case units