# CSE 403
# Lecture 4

Use Cases

Thanks to Marty Stepp, Michael Ernst, and other past instructors of CSE 403
http://www.cs.washington.edu/403/

# Use cases

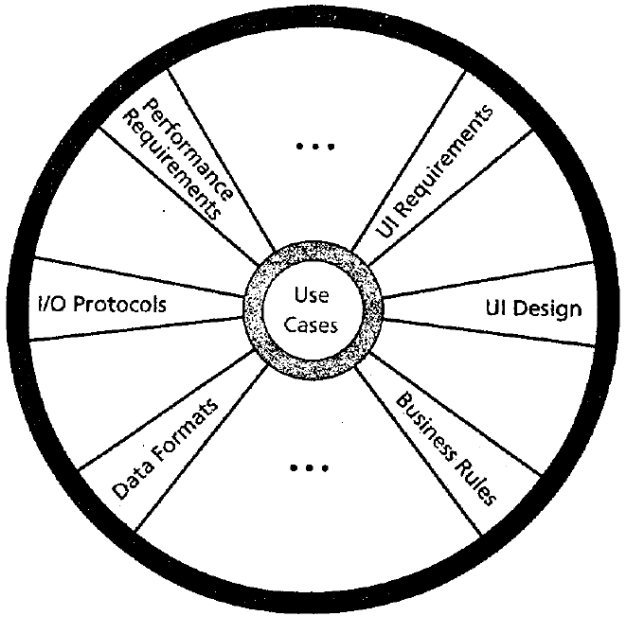
- **use case**: a written description of the user's interaction with the software product to accomplish a goal
  - interactions between an actor and the system
    - starts with a request from an actor to the system
    - 3-9 clearly written steps lead to a "main success scenario"
    - written from actor's point of view, not the system's; focuses on interactions, not internal system activities

- benefits of doing use cases?
  - helps us discover and document functional requirements
  - agreement as to the system's responsibilities
  - provides executives a skeleton for planning project priorities
  - helps provide a list of error cases to test ("extensions")

# Cockburn's template

Alistair Cockburn's suggested outline for functional requirements:

1. purpose and scope
2. terms / glossary
3. **use cases**
4. technology used
5. other
   a. development process: participants, values (fast, good, cheap?), visibility, competition, dependencies
   b. business rules / constraints
   c. performance demands
   d. security (now a hot topic), documentation
   e. usability
   f. portability
   g. unresolved / deferred
6. human issues: legal, political, organizational, training




- Cockburn says the central artifact of requirements is the **use case**.

# Use cases vs. internal features

consider the software to run a mobile phone:

**Use Cases**

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

*Point of view: user*

*("what")*

**Internal Functions**

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

*Point of view: developer / designer*

*("how")*

# Use cases and requirements

- Which of these requirements would probably be represented or mentioned directly in a use case?
  - Special deals may not run longer than 6 months.
  - Customers only become preferred after 1 year.
  - A customer has one and only one sales contact.
  - Database response time is less than 2 seconds.
  - Web site uptime requirement is 99.8%.
  - Number of simultaneous users will be 200 max.

- Answer: None!
  - Most of these are non-functional requirements, so the use cases wouldn't mention them. The user doesn't see them directly.

# Actors, stakeholders, goals

- What are **actors** and **stakeholders** in a use case?
- **actor**: anything with behavior (that "acts" on the system); can be human, external hardware or another system
  - **primary actor**: initiates interaction to achieve goal
  - **supporting actor**: performs sub-goals in use case
- **stakeholder**: anyone interested in the system
  - supplier, stock agency, vendor
  - stakeholder might not "act" in any scenario
- **goal**: action that actor wants to accomplish
  - summary (multi-sitting), user (one sitting), subfunction (partial)

# Use case exercise

- Consider a Netflix-like video rental web system:
  - A customer with an account can use their membership and credit card in the web app to order a video for rental.
  - The software can look up movies and actors by keywords.
  - A customer can check out up to 3 movies, for 5 days each.
  - Late fees can be paid at the time of return or at next checkout.

- Exercise:
  - Come up with **3-4 use case names** for this software.
    - Usually a short verb phrase that clearly states the actor's goal
  - Identify some of the **actors and stakeholders** in this system.

# Styles of use cases

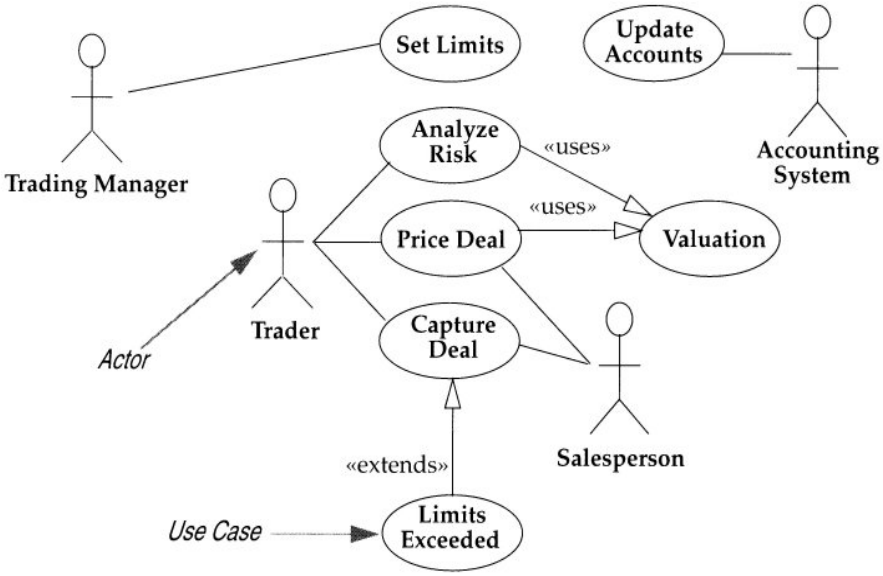
1. **Actor / goal list** (or UML use case summary diagram)
   - shows all use cases in system
2. **Informal use case**
3. **Formal use case**

Let's examine each of these in detail...

# 1. Actor / goal list

- Often shown as a list or table of actors and their goals, or a diagram of actors connected to use cases:

| ACTOR | USE CASES INITIATED |
|---|---|
| Club Member | Submit Promotion Order<br>Submit Regular Order |
| Potential Member | Submit New Subscription |
| Past Member | Submit Subscription Renewal |
| Membership Dept. | Request Membership |
| Marketing Dept. | Create Monthly Promotion<br>Create Seasonal Promotion<br>Create Subscription Program<br>Request Promotion Reports<br>Request Sales Reports |
| Services System | Send Subscription Offer<br>Send Club Promotion<br>Send Subscription Renewal |




Note: These are independent examples from two different systems.

# 2. Informal use case

- **informal use case**: a paragraph describing the scenario
- Example (for a Netflix-like video rental system):
  - Customer Loses a Disc
    The customer reports to the system that he has lost a disc. The service representative displays the customer's rental record and alerts the customer that the loss of a disc will incur a fee. The customer is asked to confirm the decision. After confirming the decision, the database will be updated to reflect the lost disc, and the customer's record is updated as well to remove the disc and add the fee. The inventory manager is notified of the event and may authorize purchase of a replacement disc.

# Extensions

- What is an "extension"?  Why are they useful?

- **extension**: A possible branch in a use case scenario, often triggered by an error or failure in the process.
  - Useful for finding edge cases that need to be handled and tested.

  - Do: Think about how every step of the use case could fail.
  - Do: Give a plausible response to each extension from the system.
    - Response should either jump to another step of the case, or end it.

  - Don't: List things outside the use case ("User's power goes out").
  - Don't: Make unreasonable assumptions ("DB will never fail").
  - Don't: List a remedy that your system can't actually implement.

# 3. Formal use case

(goal of primary actor)

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

(primary actor)

"Place an order" (User goal / Service Rep)

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

*Main scenario*:

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

...

(condition causing different actions)

(action step(s) handling those conditions)

*Extensions*:

1a. Low credit & Customer is 'Preferred':

1a1. System gives them credit anyway.

1b. Low credit & not 'Preferred' customer:

1b1. Service Rep performs Sign Up Preferred Customer scenario and accepts only prepayment.

(calling another use case)

2a. Low on stock: Customer accepts rain-check:

2a1. Service Rep reduces order to available stock level.

# Example formal use case

## Use Case 12. *Buy stocks over the web*

Primary Actor: Purchaser (user) Level: user goal
Precondition: User already has StockPurchase (SP) open and is logged in.
Minimal Guarantee: sufficient log information exists that SP can detect what went wrong.
Success Guarantees: remote web site acknowledged purchase, user's portfolio updated.

Main success scenario:
1. User selects to buy stocks over the web.
2. SP gets name of web site to use (E*Trade, Schwabb, etc.)
3. SP opens web connection to the site, retaining control.
4. User browses and buys stock from the web site.
5. SP intercepts responses from the web site, and updates the user's portfolio.
6. SP shows the user the new portfolio standing.

Extensions:
2a. User wants a web site SP does not support:
    2a1. System gets new suggestion from user, with option to cancel use case.
3a. Web failure of any sort during setup:
    3a1. System reports failure to purchaser with advice, backs up to previous step.
    3a2. Purchaser either backs out of use case or tries again.

# Formal formatting tips

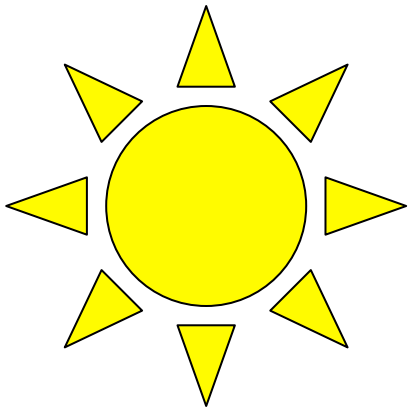
- A single use case captures a flow of events, from the triggering request from an actor to the production of results and/or return of answers from the system
- Each step is numbered sequentially
  - Alternative behaviors are labeled with the step number and alternative step or reference to another use case by name
  - Alternative steps may be indented
  - Responses by the system may be differentiated in some way such as indentation
- Be consistent with formatting style as an aid to readability
- Include summary information such as the triggering actor, pre-conditions that must be satisfied, and any post-conditions that will be true after the use case occurs

# Cockburn's 4 use case steps

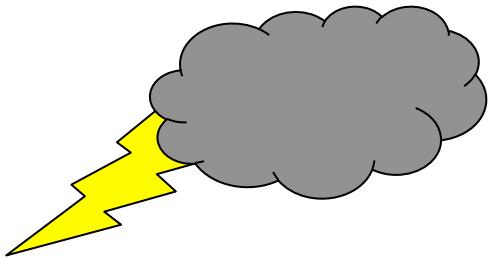
1. **Identify actors and goals**
   - What computers, subsystems, people will drive our system?
   - What does each actor need our system to do?
2. **Write the main success scenario**
   - easiest to read; everything else is a complication on this
   - capture each actor's intent and responsibility
3. **List the failure extensions**
   - usually almost every step can fail (bad credit, out of stock...)
   - note failure condition separately, after main success scenario
4. **Describe failure-handling**
   - recoverable: back to main course (low stock + reduce quantity)
   - non-recoverable: fails (out of stock, or not a valued customer)

# Success vs. failure

- "Sunny Day" scenario
- Capture the core functionality of the system under ideal conditions



- "Rainy Day" extension
- Worry about defining what could go wrong and how to handle such situations after the core functionality has been stabilized (or is as stable as it is likely to get)



# Agile "User Stories"

- Instead of Use Cases, Agile project owners do "user stories"
  - **Who** (user role) – Is this a customer, employee, admin, etc.?
  - **What** (goal) – What functionality must be achieved/developed?
  - **Why** (reason) – Why does user want to accomplish this goal?

As a [user role], I want to [goal], so I can [reason].

- Example:
  - "As a user, I want to log in, so I can access subscriber content."

- **story points**: Rating of effort needed to implement this story
  - common scales: 1-10, shirt sizes (XS, S, M, L, XL), etc.

# How much is “enough”?

- You have to find a balance between:
  - comprehensible vs. detailed
  - graphics vs. explicit wording and tables
  - short and timely vs. complete and late

- Your balance may differ with each customer depending on your relationship and flexibility