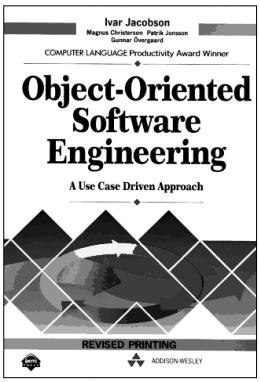
Use-Case Analysis

Use-Case Analysis

- What is it?
  - An informal, user-friendly, technique useful for *functional* requirements analysis and specification
- From where did it come?
  - Ivar Jacobson, a Swedish software engineer, now at Rational, in a *method* called OOSE (Object-Oriented Software Engineering)
- Now "part of" UML



Definition of "Use Case"

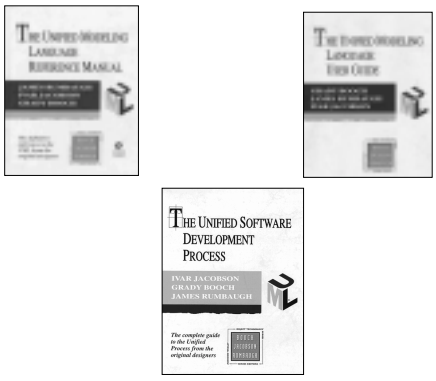
- "The specification of sequences of actions that a system, subsystem, or class can perform by interacting with outside actors"

  (*UML Reference Manual*, Rumbaugh, Jacobson, and Booch).

Purpose of a "Use Case"

- "to define a piece of behavior of a [system or subsystem or class] ***without revealing the internal structure*** of the [system]

  (UML Reference Manual, Rumbaugh, Jacobson, and Booch).



Importance of Use Cases

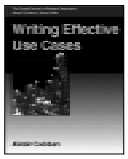
- At least one popular methodology (the Rational Unified Process, based in part on Ivar Jacobson's earlier OOSE) is said to be **Use-Case *Driven***,
- meaning that most development activities are ***traceable*** back to the use cases as defined in agreement with the user or customer.

Nonetheless

- Use cases alone do not constitute a *complete* SRS.
- For example, they focus on the functional requirements exclusively.

Use-Case Books

recommended:

not recommended:

Related Earlier Idea

- **Function-point** analysis
- Function points are the set of specific features or operations in a software product.
- Function points are used more for cost analysis than for SRS as such.
- Promoted, IFPUG (International Function Point Users' Group)

Function Point Books

"Use Case" vs. "Function Point" Opinion

- "A use case is a function that returns an observable value to an actor (object outside its context), without revealing the design structure of that function. A use case is roughly the same as a function point–a cohesive piece of functionality of the system that is visible from outside.

  Doing Hard Time, Bruce Douglass.

Other Implications

- Use cases could be used for *other* types of design, and system analysis, **not just software**.
- Once you know about them, it is hard to imagine an engineering project or business process of almost any kind starting without them.

Characteristics of Use-Case Analysis

- **Use-cases**: The specific ways in which the system is used.
- Each use-case expresses a "compete thought" or end-to-end transaction.
- A "black-box" specification; does not deal with internal structure except in possibly superficial ways.

Some Key **Components** of Use-Case Analysis

- **Actors**: Entities that use or are used by the system; *typically* people, but could *also* be other *systems* or *devices*.
- **Connections** from Actors to Use-Cases
- **Relationships** between Actors or between Use-Cases

Actors

- Actors are characterized not by the identity of the user or other, but rather by the ***role*** played by the actor.
- One person can be several different actors in different roles.
- One actor can be played (at different times) by several different persons.
- An entire **committee** could be **one** actor.
- An actor need not be a living thing; it could be another subsystem.

More on Actors

- Actors are **not part of the system** in question; they supply input to and receive output from, to the system.
- In other words, actors collectively define the **environment** of the system.

Minimum Requirement for a Use Case

- Verbal description

Common Components of a Use Case

- Name
- Symbolic label
- List of actors
- Initiator
- Verbal description

Initiator

- The initiator of a use case is the actor that starts the flow of events.

Brief Use-Case Description

label for this use-case *name of this use-case*

- OCI: Order from catalog
  - **Actors**: customer, sales rep., shipping dept.
  - **Initiator**: customer
  - **Description**: Customer calls to order items from the catalog. The sales rep. identifies the item numbers, verifies that the items are in stock, and confirms the order with the customer, giving him the order number. The sales rep. then forwards the order to the Shipping dept.

Flow of Events

- Could be more readable for the description to be an enumerated list of events, e.g.:
  1. Customer calls to order from catalog.
  2. Sales representative identifies item numbers.
  3. Sales representative verifies stock.
  4. Sales representative confirms order.
  5. Sales representative gives order number to Customer.
  6. Sales representative passes order to Shipping.

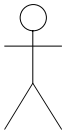
Flow-of-event descriptions could contain *iteration*

- An order could contain multiple items. In this case, the event flow should show something like:
  - ***For each*** item to be ordered:
    - Sales representative checks catalog number.
    - Sales representative verifies stock.
    - Sales representative records item.
- Similarly, flow of events could contain conditional (if-then-else) behaviors.

Use Case Diagrams

- For visualization of use cases
- Don't tell the whole story
- Useful in brainstorming
- Used in software tools, such as:
  - Rational Rose
  - iLogix Rhapsody

Icon for an Actor



Note: Actors are typically drawn in this "anthropomorphic" way even when the actors aren't people.

Examples of Actors

Customer

Shipping Dept.

Alternate Actor Icons in UML

Customer

Visual Icon

«actor»
Customer

Textual Stereotyping

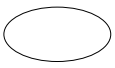
«actor»
Customer

Textual & Visual Stereotyping

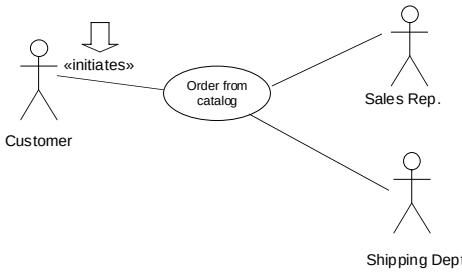
UML Use of "Stereotypes"

- The « … » notation called "guillemets", (used for quotes in French, Italian, and Spanish)
- These usually indicate the name of a "stereotype", defined as an *informal extension* of the UML.

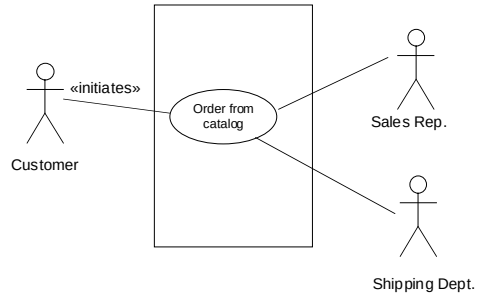
Icon for a Use Case



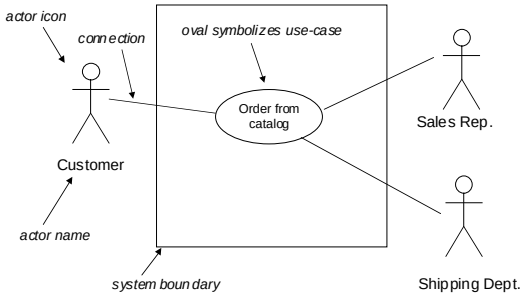
Noting Initiator



A simple use-case for a mail-order catalog business



Symbology for a simple use-case



Class Exercise

- Identify several other possible use-cases in the catalog-order enterprise.
- For each use case, indicate the actors, initiator, and flow of events.

Some Possibilities

- Cancel order
- Modify order
- Inquire status of order
- Fill back order
- Request product return authorization
- File complaint
- Supervisor monitoring order
- Inform customer of current specials
- Request catalog
- Request to be taken off mailing list

More Possibilities

- Run sales report
- Run back-order report
- Run customer history
- Update catalog
- Get product popularities
- Introduce sale items
- Remove sale items

Steps in Use-Case Analysis

- Identify system boundaries
- Identify actors:
  - Recall: an actor is an entity playing a particular role with respect to the system.

Steps in Use-Case Analysis (cont'd)

- Identify use cases themselves:
  - Every use case has at least one actor.
  - A specific actor initiates the use case.
  - The same actor may participate in multiple use cases, as initiator in some and not in others.
- Create the description including flow of events
- Identify clarifying ***scenarios*** where helpful
- Provide additional information (see later)

Scenarios of a Use Case

- A "scenario" is a *single* path through the event flow. For example, if there is a conditional part, only one branch is taken in the scenario.
- Obviously we can't always enumerate all the scenarios; there might be an infinite set of them. If the use case involves iteration, only a finite number of iterations are used in the scenario.

Scenarios (continued)

- Often there will be a "principal" scenario, and several secondary variations.

A Catalog Order Scenario (1 of 3)

- Alice calls company.
- Bert answers the telephone.
- Alice indicates she wishes to place an order.
- Bert asks how the order will be paid.
- Alice indicates via credit card.
- Bert asks for the card number,billing address, and expiration date.
- Alice provides the above info.

A Catalog Order Scenario (2 of 3)

- Bert asks for the first item.
- Alice responds with first item.
- Bert asks for quantity of first item.
- Alice responds with quantity of first item.
- Bert records first item with quantity.
- Bert asks for second item.
- Alice responds with second item.
- Bert indicates second item out of stock; does Alice wish it to be back ordered?
- Alice declines to order item.

A Catalog Order Scenario (3 of 3)

- Bert asks for third item.
- Alice responds that there are no more items.
- Bert asks for shipping address.
- Alice indicates that it is the same as the billing address.
- Bert informs Alice of expected shipping date and provides order number.
- Bert thanks Alice.
- Alice hangs up.
- Bert transmits order to Shipping dept.

Scenarios and Exceptions

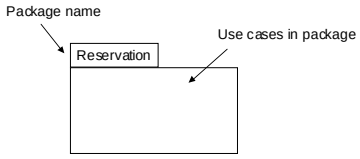
- One possible use of scenarios is in demonstrating representative exceptional or “what if” behaviors (as opposed to the principal behavior).
- **Example**: In the catalog order use case, what if the customer hangs up or connection is lost in the middle of the dialog?

Scenario Types (Bruegge)

- **Visionary** scenario: Describes future scenario
- **Evaluation** scenarios: Describe user tasks against which system is evaluated
- **Training** scenarios: Used for tutorial purposes
- **As-is** scenarios: Describe current situation (during reengineering)

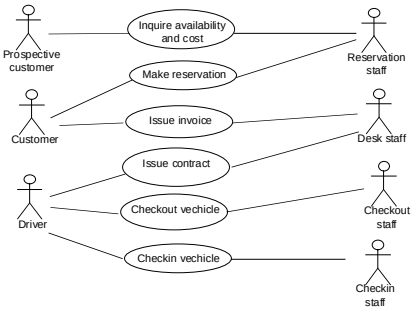
UML **Packages**

Used for Grouping; Could be Used to Group Use Cases

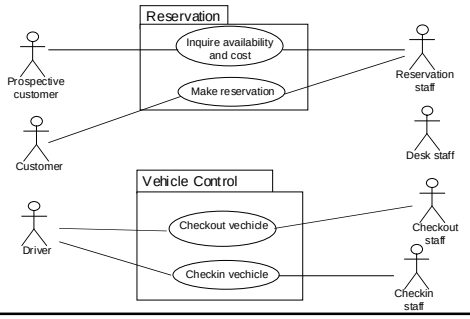


Car Rental Example

How might the use cases be packaged?



Car Rental Example with Two Packages



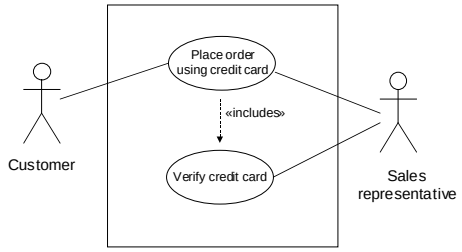
Use of Packages

Packages may be used in the transition to a design, and ultimately coding.

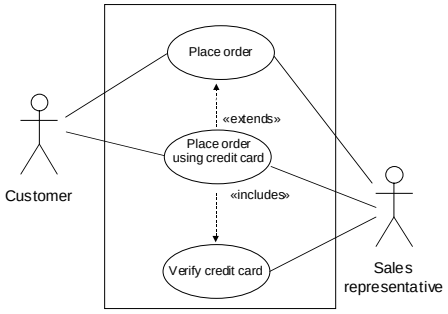
However, these connections would not normally be part of the initial discussion with the customer.

Relations between Use Cases

Inclusion among use-cases



Extension among use-cases

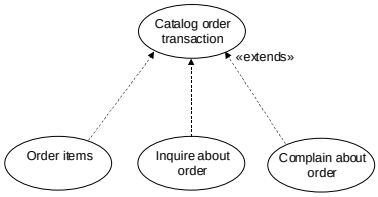


extends vs. *uses*

- **«includes»** means this use-case makes use of another use-case, as if a kind of *subroutine*. This allows us to not have to repeat the included use-case in the description of the including use-case.
- **«extends»** means that this use-case is a *specialization* of another use case.

Note: «includes» was formerly called «uses».

«extends» can be used to impart a hierarchical **abstraction structure** to use cases



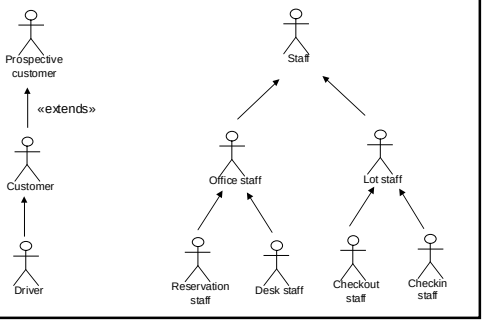
Extension in this sense is analogous to class extension in object-oriented programming.

Options of a use case

- **Example**: During order processing, the sales representative offers to tell the customer about current specials.
- Such options should be mentioned as an annex to the other use case items.

Actor Hierarchies

are possible, similar to extensions



Caution about Structuring Use-Cases

- Use-case structuring is obviously *analogous* to structuring in object-oriented systems.
- However, one should ***not*** infer that use-case structure implies anything about *internal* structure.

Use-cases vs. Requirements

- A use-case describes one “unit” of functionality.
- A single informally-specified functional requirement could translate into multiple use-cases.
- A single use-case could also be involved in satisfying multiple requirements.

Use-cases vs. Requirements (cont’d)

- **Collectively**, the use-cases ideally account for all of the desired functional requirements.
- Non-functional requirements may *annotate* use-cases, but don’t get represented as use-cases directly.

Further Components of Use Cases

Goals

- A goal describes the **higher purpose** of the execution of the use case.
- Example: Goal for catalog order: A customer wishes to order products from the company.

Pre- and Post-Conditions

- Some use-cases are not meaningful at arbitrary times, but instead only when the system is in a state with certain properties. Such properties are called ***pre-conditions***.
- Similarly, the use-case might leave the system in a state known to satisfy one or more ***post-conditions***.

Example:

- For the car-rental enterprise, the use case "checkin vehicle" has the pre-condition
  vehicle is rented to driver
  and the post-condition:
  vehicle is on site
  & vehicle is not rented to driver.
- For the use-case "checkout vehicle", these conditions are reversed.

Invariants

- A condition that is a pre- and post-condition for all use cases is called an ***invariant***.
- Example: Total vehicles = vehicles rented + vehicles available + vehicles in repair + vehicles in scrap.

Optional Triggers

- A *trigger* is an event that causes the use case to be run.
- Example: A catalog order is triggered by a phone call.
- This is similar to a pre-condition, but is a dynamic event rather than a condition.

Exceptions

- If a use case cannot be completed as described, an *exception* is said to occur.
- The description can indicate aspects of the state and output in such cases.

Alternative to Exceptions

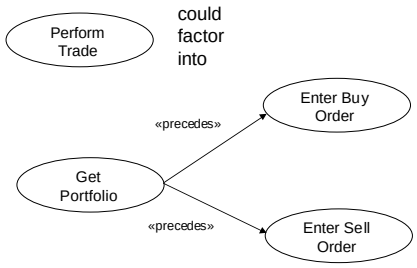
- A use case may be allowed **explicit success and failure outcomes**, each with its own post condition.

Precedence among Use-Cases

- When one use case is used to establish a pre-condition for another, the two may be linked by «precedes».
- One use of precedence is to *factor* a use-case into sub-cases, to avoid repetition among different sub-cases.

Precedence Example

Stock-trading example:



Some of the

Top 10 Use-Case Pitfalls

by Susan Lilly, Software Development Magazine, Jan. 2000
http://www.cs.uwf.edu/~italbot/cen5990/lib/use_case_pitfalls.html

- The use cases aren't written so the customer can understand them.
- The system boundary is undefined or inconsistent.
- The use cases are written from the system's (not the actors') point of view.
- The actor names are inconsistent.
- A use case doesn't correctly separate actors based on functional entitlement.
- The use-case specifications are too long or confusing.

More Use-Case Advice (Larry Constantine and others)

- Write in the active voice.
- Pair responses with the events that invoke them.
- Identify **domain objects** that clearly are part of the application context (such as "catalog", "inventory", "fleet" (of automobiles)). [A domain dictionary could be used.]

Example

Use Cases for the Meeting Scheduling System

Bob Keller
January 2001

List of all Use Cases by Name

- Add Indication of Unavailability
- Add Indication of Availability
- Check Availability for a Meeting
- Cancel Meeting
- Cancel All Meetings
- Move Meeting to a Different Time
- Check Whether a Meeting is Movable
- Check Availability for a Meeting and Schedule if Positive
- Determine Possible Times for Meeting
- Modify the Participants in a Meeting
- Show all Meetings this Caller has Called
- Cancel Participation in a Scheduled Meeting
- Show Participant's Schedule
- Define a Participant Set with a Label
- Modify a Participant Set Having a Given Label
- Dissolve a Participant Set Having a Given Label
- Show the Defined Sets
- Authorize a Participant to Call Meetings or Define Sets
- Initialize the System
- Provide Help in Use of the System

Template: Label: Name

- Goal
- Actors
- Initiator
- Description
- Pre-conditions
- Post-conditions
- Options, if present
- Scenario, if helpful in clarifying

Glossary

- **Participant**: An actor who can participate in some meetings
- **Meeting caller**: A participant who can call a meeting
- **Participant Set**: A mathematical set consisting of participants, together with an indication of essential, desirable, or optional for each.
- **Set definer**: A participant who can define a set and gives it a symbolic name.
- **Administrator**: An actor who can initialize the system and who put other actors in the categories of meeting caller, set definer and administrator.

AddUnavailability: Add Indication of Unavailability

- Goal: To enable a participant to give an indication of unavailability
- Actors: Participant
- Initiator: Participant
- Description: The user indicates that he/she is unavailable during a specific time interval. The system notes this fact.
- Pre-conditions: none
- Post-conditions: Meetings may be not scheduled with the user as a participant during the interval.
- Options: The unavailability can be indicated as one-time or recurrent. In latter case, the first and last date, and the repeat interval, can be specified.
- Scenario
- Note: The default state for a user is available until unavailability is specified.

AddAvailability: Add Indication of Availability

- Goal: To enable a participant to add an indication of availability
- Actors: Participant
- Initiator: Participant
- Description: A particpant indicates that he/she is available during a specific time interval. The system notes this fact.
- Pre-conditions: none
- Post-conditions: A meeting may be scheduled with the participant as a participant during the interval.
- Options
- Scenario
- Note: This effectively undoes any prior expression of unavailability at the indicated time.

CheckAvailability: Check Availability for a Meeting

- Goal: To enable a user to check the feasibility of a meeting at a specific interval with a specified set of participants
- Actors: Meeting caller
- Initiator: Meeting caller
- Description: The meeting caller asks whether a set of users is available for a meeting during a specific time interval.
- Pre-conditions:
- Post-conditions:
- Options: If availability is positive, the meeting can be scheduled (see SetMeetingPendingAvailability)
- Scenario

CancelMeeting: Cancel Meeting

- Goal: To enable a meeting caller to cancel a meeting previously scheduled
- Actors: Meeting caller, participants
- Initiator: Meeting caller
- Description: The meeting caller indicates that a previously scheduled meeting is cancelled.
- Pre-conditions: The identified meeting exists.
- Post-conditions: The identified meeting no longer exists.
- Options: Some or all sessions of a recurrent meeting can be cancelled.
- Options: MoveMeeting, CancelAllMeetings
- Scenario

CancelAllMeetings:
Cancel all Meetings Called in an Interval

- Goal: To enable a meeting caller to cancel all meetings he/she called in a given interval
- Actors: Meeting caller, participants
- Initiator: Meeting caller
- Description: The meeting caller indicates that all meetings scheduled in a given interval are cancelled.
- Pre-conditions: The identified meetings exist.
- Post-conditions: The identified meetings no longer exists.
- Options:
- Scenario

MoveMeeting :
Move Meeting to a Different Time

- Goal: To enable a meeting caller to move a meeting to a different time
- Actors: Meeting caller, participants
- Initiator: Meeting caller
- Description: The meeting caller indicates that a previously scheduled meeting is moved to a different time. The participants involved are rescheduled, if possible, and notified. The caller is notified of the outcome.
- Pre-conditions: The identified meeting exists.
- Post-conditions: The meeting is moved to a different time.
- Options : Some or all sessions of a recurrent meeting can be indicated.
- Scenario

CheckMovability :
Check Whether a Meeting is Movable

- Goal: To enable a meeting caller to assess whether a meeting can be moved, without losing participants
- Actors: Meeting caller
- Initiator: Meeting caller
- Description: The meeting caller asks whether a previously scheduled meeting is movable to a specified time.
- Pre-conditions:
- Post-conditions:
- Options : Some or all sessions of a recurrent meeting can be indicated.
- Scenario

SetMeetingPendingAvailability :
Check Availability for a Meeting and Schedule if Positive

- Goal: To enable a user to check the feasibility of a meeting at a specific interval with a specified set of participants
- Actors: Meeting caller, participants
- Initiator: Meeting caller
- Description: The user requests a listing of times at which a meeting with a specified set of participants or labeled participant set could be scheduled.
- Pre-conditions: If the meeting is schedulable, no meeting with the generated serial number exists.
- Post-conditions: The meeting is set at the time indicated, if possible, and the invitees are scheduled accordingly. The invitees are then notified. The meeting caller is notified of the outcome either way. If the meeting is schedulable, a meeting with the generated serial number exists.
- Options: The meeting can be declared as one-time or recurrent. In latter case, the first and last date, and the repeat interval, can be specified. The recurrent meetings are given serial nos S.1, S.2, ... where S is the overall serial number for this meeting.

FindMeetingAvailability :
Determine Possible Times for Meeting

- Goal: Enable a user to find a time for a meeting with a specified set of participants.
- Actors: Meeting caller
- Initiator: Meeting caller
- Description: The user requests a listing of times at which a meeting with a specified set of participants or labeled participant set could be scheduled.
- Pre-conditions:
- Post-conditions:
- Options: The meeting can be declared as one-time or recurrent. In latter case, the first and last date, and the repeat interval, can be specified.
- Scenario

ModifyParticipants:
Modify the Participants in Meeting

- Goal: Enable a meeting caller to alter the set of participants in a meeting.
- Actors: Meeting caller, participants
- Initiator: Meeting caller
- Description: The meeting caller indicates a modification to the participants in a meeting. The affected participants are notified. The caller is notified of the result, including any participants who are unavailable. (If participant sets were used, these are not modified.)
- Pre-conditions: The specified meeting exists.
- Post-conditions: The meeting participants are modified as indicated.
- Options
- Scenario

ShowMeetingsCalled:

Show all Meetings this Caller has Called

- Goal: Enable a meeting caller to show all meetings that have been called.
- Actors: Meeting caller
- Initiator: Meeting caller
- Description: The meeting caller requests to see all meetings in effect during a given interval.
- Pre-conditions:
- Post-conditions:
- Options: The caller can optionally request listing of participants.
- Scenario

CancelParticipation:

Cancel Participation in a Scheduled Meeting

- Goal: To enable a participant to indicate non-participation in a specific meeting, even if availability was indicated.
- Actors: Participant, meeting caller
- Initiator: Participant
- Description: The participant indicates that he/she will not attend a previously scheduled meeting.The caller of the meeting is notified in the case that the participant was essential.
- Pre-conditions: The participant is scheduled for the meeting.
- Post-conditions: The participant is not scheduled for the meeting.
- Options: If availability is positive, the meeting can be scheduled (see SetMeetingPendingAvailability)
- Scenario

ShowSchedule:

Show Participant's Schedule

- Goal: To enable a participant to determine the meetings at which he/she has been scheduled.
- Actors: Participant
- Initiator: Participant
- Description: The participant requests a list of meetings at which he/she has been scheduled in a given time interval.
- Pre-conditions:
- Post-conditions:
- Options
- Scenario

Define Set:

Define a Participant Set Using a Label

- Goal: To enable a meeting caller to refer to a set of participants by a single name, which can be used in other use cases.
- Actors: Set definer, participants
- Initiator: Set definer
- Description: The set definer defines a set of participants, each with a specification of essential, desirable, or optional, giving the set a symbolic label. The participants are notified of the creation of the set.
- Pre-conditions:
- Post-conditions: A set with the indicated label exists.
- Exception: A set with the indicated label already exists.
- Options
- Scenario

Modify Set:

Modify a Participant Set Having a Given Label

- Goal: To enable a set definer to modify the contents of a given set.
- Actors: Set definer
- Initiator: Set definer
- Description: The set definer specifies modification of a given Particpant Set.
- Pre-conditions: A set with the label exists.
- Post-conditions: The set is modified as specified.
- Exceptions: No set with the label exists. The proposed modification doesn't agree with the set constituency.
- Options
- Scenario

DissolveSet:

Dissolve a Participant Set Having a Given Label

- Goal: To enable a set definer to dissolve a set no longer used.
- Actors: Set definer
- Initiator: Set definer
- Description: The set definer dissolves an existing list. Participants on are notified.
- Pre-conditions: A set with the label exists.
- Post-conditions: No set with the label exists.
- Exception: No set with the label exists.
- Options
- Scenario

ShowSets:
Show the Defined Sets

- Goal: To enable a set definer to see the defined sets.
- Actors: Set definer
- Initiator: Set definer
- Description: The set definer requests a list of defined sets, which is then provided.
- Pre-conditions:
- Post-conditions:
- Options
- Scenario

Authorize/Unauthorize:
Authorize a Participant to Call Meetings or Define Sets

- Goal: To enable participants to have special roles.
- Actors: Administrator
- Initiator: Administrator
- Description: An administrator adds or removes privileges such as those of meeting caller and set definer for participants.
- Pre-conditions: The identified participants possess or lack certain privileges.
- Post-conditions: The identified participants possess or lack certain privileges.
- Options
- Scenario

Initialize:
Initialize the System

- Goal: To enable the system to be reinitialized.
- Actors: Administrator
- Initiator: Administrator
- Description: An administrator (re-)initializes the system, removing all meetings, defined sets, and privileges.
- Pre-conditions:
- Post-conditions: There are no privileged actors other than the administrator, no meetings, and no defined sets.
- Options
- Scenario

Help:
Provide Help in Use of the System.

- Goal: To enable actors to determine how to use the system without consulting a manual.
- Actors: Any actor
- Initiator: Any actor
- Description: The actor requests help. The system replies with a help message.
- Pre-conditions:
- Post-conditions:
- Options: Help on specific topics
- Scenario

CS 121

Assignment 2

Due Monday, 29 January 2001

Develop a Set of Use Cases

- Develop a set of use cases for an auction web-site application.
- Give descriptions using the attached template, as well as diagrams.

Use-Case Template

Use the following template for use cases:

- Label
- Name
- Goal
- Actors
- Initiator
- Description
- Pre-conditions
- Post-conditions
- Options (if present)
- Scenario, if helpful in clarifying

Background Information

- If you are unfamiliar with auctions, you may wish to surf one or more of the following:
  - www.ebay.com
  - s1.amazon.com
  - www.ubid.com
- The following glossary might be useful:
  http://pages.ebay.com/help/basics/g-index.html

Purpose

- The purpose of this exercise is **not** to reverse-engineer any **particular** auction site.
- Rather it is to think through a set of consistent use cases that completely covers the minimal requirements.

Nature of Actors

- The number of actors in an auction application is unlimited. So it is advisable to use a single actor as a typical seller, a single actor as a typical bidder, etc.

Minimal Requirements (1) Seller

- It must be possible for a seller to list items, to find out the set of current bids, to be notified of the outcome of an auction, to cancel an auction (provided that no bids have been made), to receive payment for an item, to notify the site that the item has been shipped, to have a complaint levied on him/her.

Minimal Requirements (2) Buyer

- It must be possible for a buyer to bid for items, to be notified of being outbid, to be notified of the outcome of an auction, to engage proxy bidding, to find out items on which he/she currently has bids, to pay for an item in the event of winning the auction, to query information about the seller, and to file a complaint.

Minimal Requirements (3) Other

- It must be possible for the administrator of the site to set listing fees and commission rates.
- It must be possible for a browser (person) to search for items, and browse categories of items.
- It must be possible for any kind of user to register as either a buyer or seller, which results in the user having a password. Also, any registered user can unregister. Any user can add or withdraw from his/her account and inquire as to account status.

Traceability Matrix

- Make a Traceability Matrix, which lists each of the requirements and lists the use cases that cover those requirements.

Additional Points on Use Cases

Jacobson, in OOSE

- Jacobson suggests that Use Cases *become* Objects in the design.
- While this may happen, it is probably not advisable that use cases necessarily become objects.
- This would be dictating internal structure in the specification.

Do *Not* use Use-Cases to Fully Decompose into a **Design**

- Factoring should be used to ***simplify the description*** of use-cases.
- Avoid the temptation of making use-case decompositions into design.
- Use-cases are ***customer language***, ***not*** design language or pseudo-code. They describe *what*, not *how.*
- There are other tools that are better-suited to the design phase.

Uses of Use-Cases **across** Development Phases

(Bruce Douglass, "Doing Hard Time")

- Analysis phase:
  - Suggest large-scale partitioning of the problem domain
  - Provide structuring of analysis objects (i.e. actors and sub-systems)
  - Clarify system and object responsibilities
  - Capture new features as they are added
  - Validate analysis model

Uses of Use-Cases across Development Phases (Bruce Douglass) (cont'd)

- Design phase:
  - *Validate* the elaboration of analysis models in the presence of design objects
- Coding phase:
  - *Clarify* purpose and role of classes for coders.
  - Focus coding efforts
- Testing phase:
  - Provide test scenarios for *validation*
- Deployment phase:
  - Suggest iterative prototypes for spiral development

Refinement of Use Cases

- These ideas are from Use Cases: Requirements in Context, Daryl Kulak and Eamonn Guiney, ACM Press, 2000.
- Four iterative levels for specifying use cases:
  - **Façade**: Outline and high-level descriptions
  - **Filled**: Broader and deeper descriptions
  - **Focused**: Narrowing and pruning
  - **Finished**: Touch-up and fine-tuning
- See the reference for example worked out at all levels.

Façade Use-Case Components

- Name
- [Goal]: I added this.
- Summary
- Basic course of events

Filled Use-Case Components

- Name
- [Goal]
- Summary
- Basic course of events
- Alternative paths
- Exception paths

Focused Use-Case Components

- Name
- [Goal]
- Summary
- Basic course of events
- Alternative paths
- Exception paths
- Extension [Option] points
- Trigger
- Assumptions
- Preconditions
- Postconditions
- Related ***business rules***

Business Rules

- **Business rules** are requirements that represent ***constraints*** on behaviors, rather than behaviors themselves.
- Examples:
  - All transactions in U.S. Dollars.
  - Valid operators license required in order to rent.
  - Late-fee assessed after the second week of the semester.

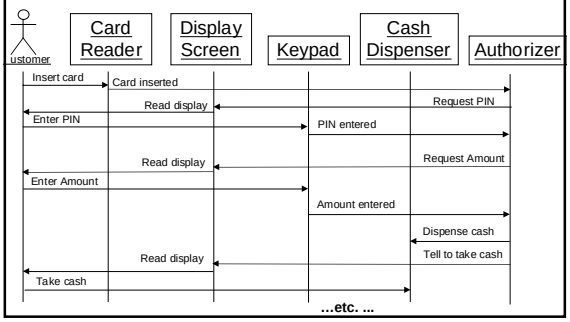
Finished Use-Case Components

- Same components as in the Filled iteration, just more polished.

UML Ways of Clarifying Complex Behaviors in Use Cases

- These are more technical and may be more appropriate in the ***design*** phase. However, sometimes they can clarify a use case:
  - **Sequence diagram**: shows messages between actors and sub-systems
  - **Collaboration diagram**: a sequence diagram organized as a directed graph rather than as a linear sequence of messages.
  - **State chart**: Elucidates behavior in terms of properties of state
  - **Timing diagram**: a sequence diagram with a time metric applied to the sequence dimension

Sequence Diagram for an ATM Withdrawal Use Case



Collaboration Diagram