Professor Hausi A. Müller PhD PEng FCAE
Department of Computer Science
Faculty of Engineering
University of Victoria

www.engr.uvic.ca/~seng321/
courses1.csc.uvic.ca/courses/201/spring/seng/321

SENG 321 Calendar

Deliverable S3a	Fri, Mar 18	S3a Technical Design Spec	15% of project
Deliverable S3b	Tue, Mar 22	S3b Manual	10% of project
Quiz 3: Use cases	Wed, Mar 23	In class	2% of course
Deliverable C3	Thu, Mar 24	C3 feedback on S3a&S3b	10% of project
Easter break	Fri-Mon, Mar 25-28	Fri, no class	
Deliverable S4	Mar 29-Apr 1	S4 project demo (in TWF classes and Tue lab; no lab on Thu)	10% of project
Deliverable C4	Fri, Apr 1	C4 feedback on S4	5% of project
Last Day of Classes	Fri, Apr 1		
Final Exam	Sat, Apr 16	19:00-22:00 ECS 125	35%

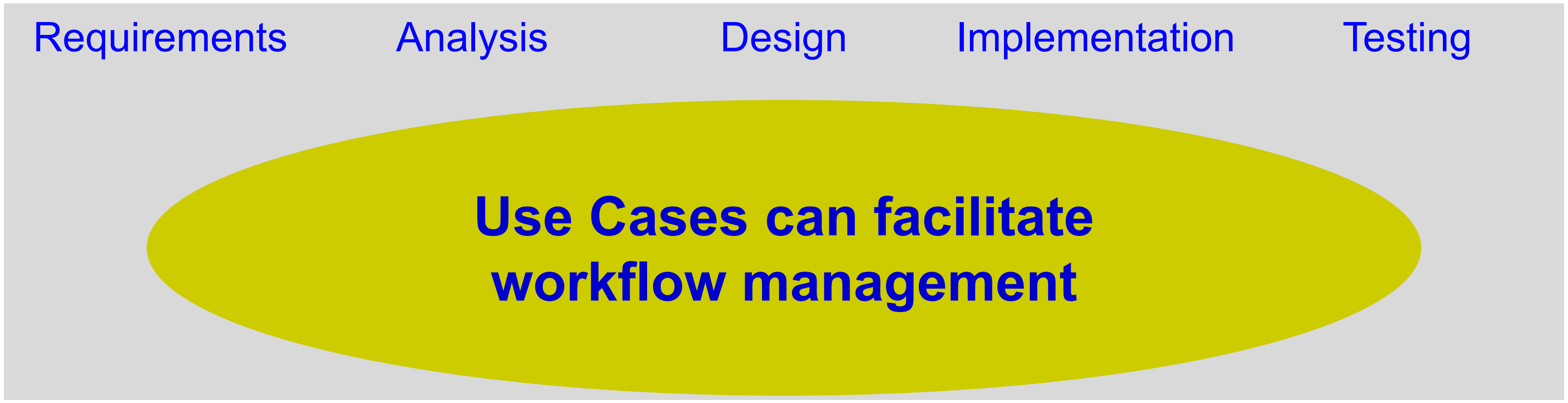
Announcements

- Fri, March 18
  - S3a due
  - Detailed technical design spec
- Tue, March 22
  - S3b due
  - User manual due
- Fri, March 25
  - Good Friday, no class
- Tue/Wed/Fri, March 29/30, April 1
  - In class and Tue lab demos
  - No labs on Thu
  - 3 presentations per hour
  - 15 mins per presentation

Final Exam

- Sat, April 16
- 19:00-22:00
- ECS 125

Use Case Driven Software Development



Benefits of Use Cases

- Developing use cases is a simple technique to
  - Identify the tasks to be performed
    - Both at macro and micro level
  - Identify the major actors and how they interact with system
  - Clarify who is responsible for what
  - Brings out hidden assumptions and ambiguities at review time
  - Clarify “what-if”s, and ensure all possible bases are covered.
  - Identify and clarify system-level test cases

What Project Areas Are Improved with Use Cases?
http://www.evanetics.com/Articles/ar_usecases/uc_areasimproved.htm

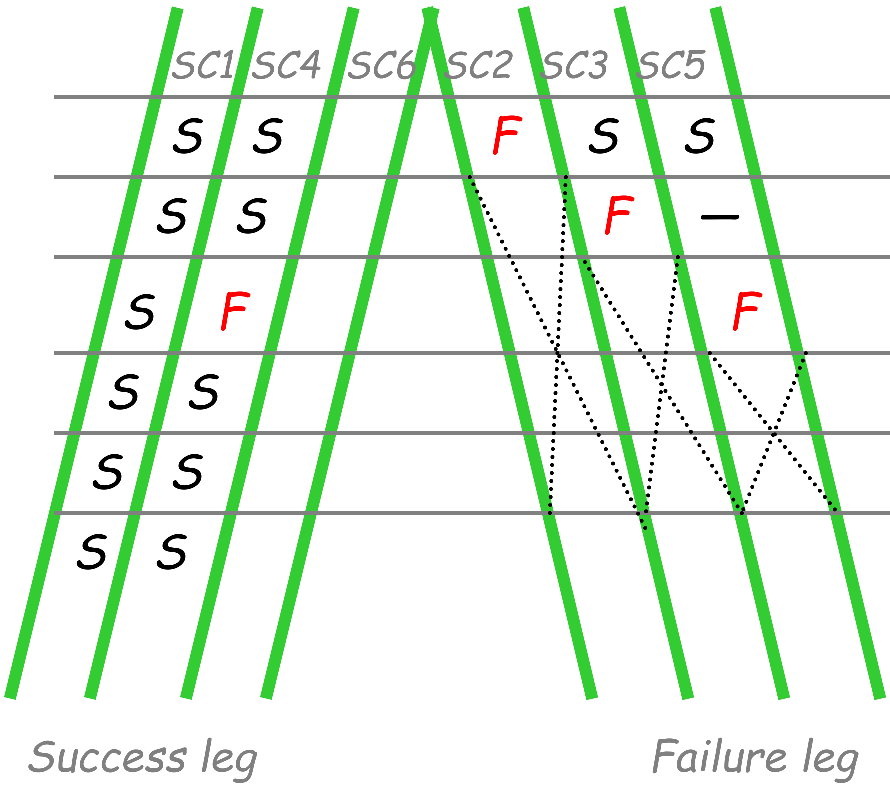
Scenarios

- A scenario is one full execution path through a use case
  - Typically, each step in a use case may have variations and error conditions.
    - A scenario traces one path from start until ultimate success or failure.
      - Path 1: a customer tries to rent a video; has no overdue videos or fines; completes the rental.
      - Path 2: a customer tries to rent a video, but has outstanding overdue fines; must first pay fines; then complete the rental.
  - A full use case comprises the set of all possible scenarios from start to finish.
  - Representation of scenarios
    - Text lists, HTML forms, or UML diagrams

Final exam question
What is the difference between a use case and a scenario?

Uses Cases and Scenarios

- A use case is a collection of success and failure scenarios describing a primary actor using a system to support a goal
- Guru Cockburn says: Think of stripes down a trouser leg.



UC7: Order Books

i. Client requests book order

ii. Check book in catalog

iii. Check customer credit

iv. Check inventory

v. Process payment

vi. Ship books.

S	Success
F	Failure
—	Don't care

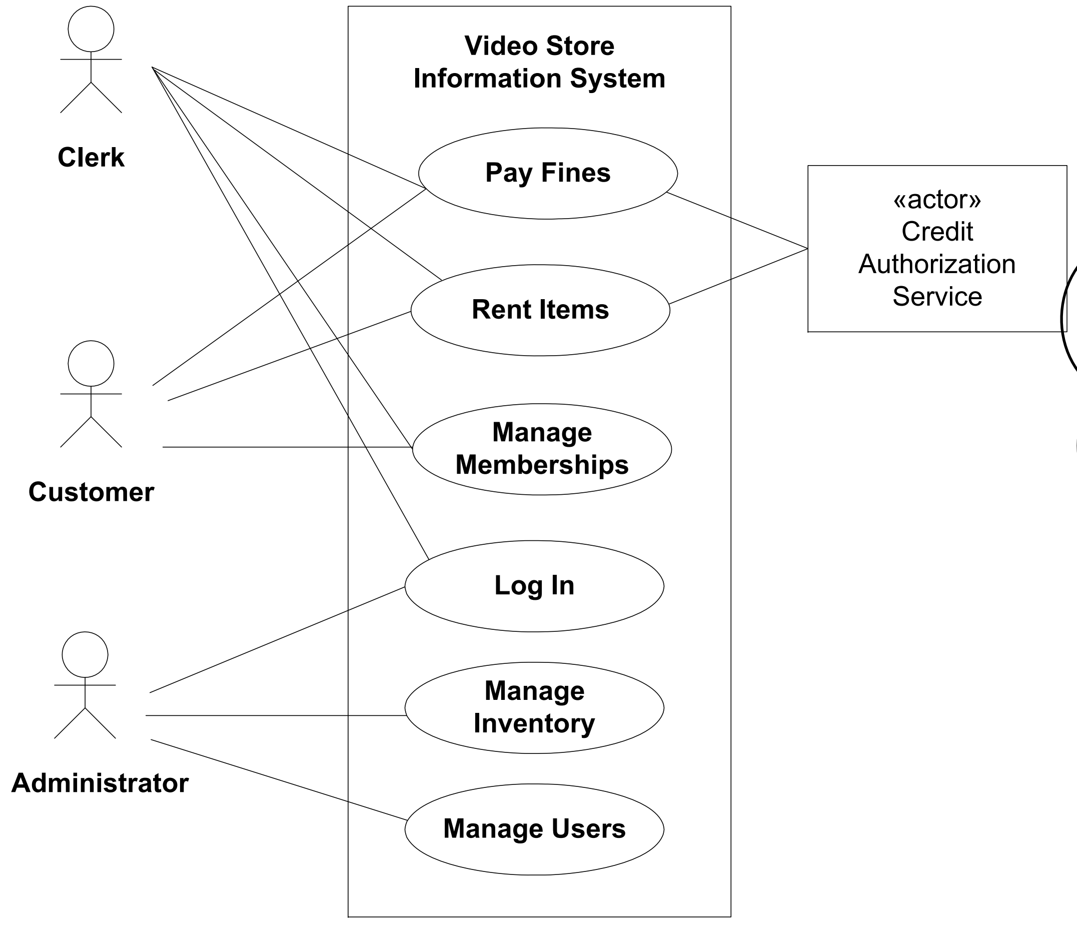
Not all Failures are Fatal

- Many failures are recoverable!
- This is a simple example where goals have obvious success or failure values.
  - In more complicated situations, you may have multiple possible values, complicated if- and case-statements, or loops.
  - For these cases, textual lists and UML sequence diagrams may not be enough;
    - Could use UML activity diagrams (flow charts) to illustrate scenarios

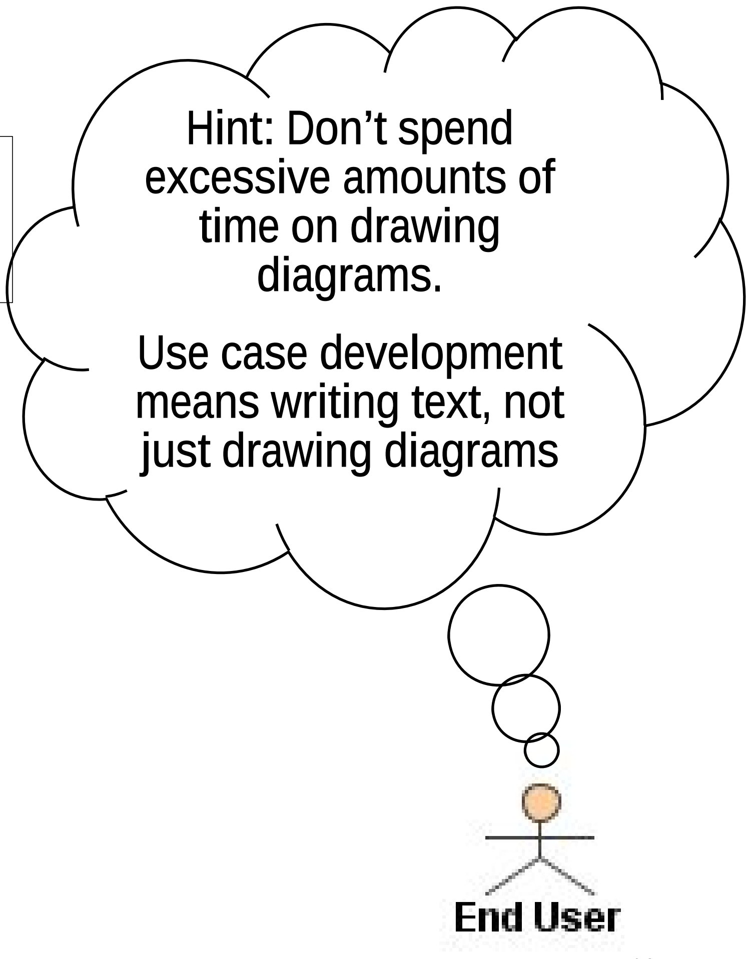
Use Cases and UML Use Case Diagrams

- Fundamentally, use cases are text, not diagrams.
  - **Use case analysis is a *writing* effort, not a drawing effort ☺**
- But drawing a UML use case diagram provides a context for:
  - Identifying and indexing use cases by name
  - Creating a context diagram
  - Providing overviews of use case sets

Context of Use Cases



[Larman]

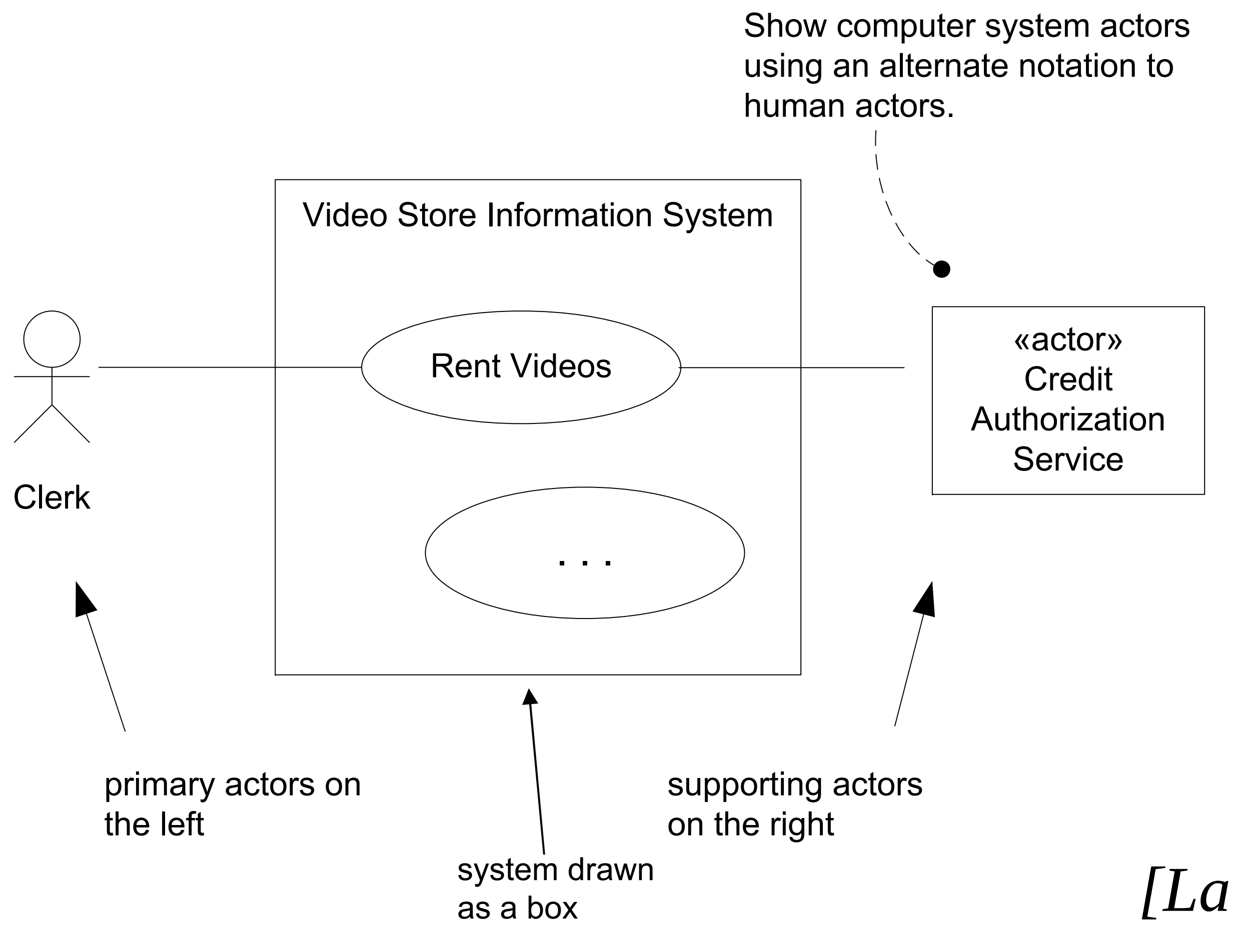


Hint: Don't spend excessive amounts of time on drawing diagrams.
Use case development means writing text, not just drawing diagrams
End User

Use Case Diagrams

- Describes the set of all use cases graphically
- Models the system’s top-level functionality and environment
- Context diagram
  - Use cases: requirements in context
- System drawn as a box
  - Can collect related use cases into packages inside the box
- Shows which actors involved in which use cases
  - Primary actors on left; supporting actors on right
  - Use a difference visual rep. for non-human actors (plus «actor» stereotype)
- [Sometimes] shows relationships between use cases
  - «**`includes`**» and «**`extends`**»

UML Use Case Diagrams


Show computer system actors using an alternate notation to human actors.
Video Store Information System
Clerk
Rent Videos
. . .
«actor» Credit Authorization Service
primary actors on the left
system drawn as a box
supporting actors on the right

[Larman slide]

«`includes`» and «`extends`»

- UC1 «**`includes`**» UC4
  - «**`includes`**» is used for "services" common to several use cases for example: *QueryBlood* used by OrderBlood
  - Like procedure call; "control" returns to UC1 at the "inclusion point" after UC4 is "executed"
- UC7 «**`extends`**» UC5
  - «**`extends`**» used for important variations
  - At extension point in use case UC7, "control" is transferred to UC5 and does not return.
- Fowler and others recommend against using these UML features
  - It encourages you to get too complicated too quickly
  - Stick to simple textual descriptions instead

Use Case Descriptions

- Use cases are fundamentally textual!
  - Use templates or predefined structures
- Possible formats
  - Brief use case
    - Terse, one-paragraph summary, usually just documenting the main success scenario
  - Casual use case
    - Informal, multi-paragraph format, covering various scenarios
  - Fully dressed use case
    - An elaborate format, with all steps and variations written in detail, covering most scenarios in detail

A Brief Use Case: Rent a Video

- A customer arrives with video store to rent. The Clerk enters the customer’s ID, and each video ID. The system outputs information on each. The Clerk requests the rental report. The system outputs it, which is given to the customer with the rented videos.

A Fully Dressed Use Case: Buying a Book Online

Name: Buy a book online

Use Case Number: UC32

Authors: John Doe

Event: Customer requests to buy one or more books. The choice of books is passed as the input.

System: Customer and vendor computers with web applications that implement online book selling

Actors:

- Customer (initiator)
- Credit-card authorization service
- Bookseller

Overview: This use case captures the process of purchasing one or more books from an online book seller.

References: R23, R34, and R45.

Related Use Cases: UC11

Typical Process Description

Actor Action	System Responsibility
1. Customer submits a selection of books he or she wants to buy.	
	2. System checks if the customer has already identified himself. If customer is not identified, see UC11 (Shopping Cart Set Up).
	3. System adds books to the Shopping Cart.
	4. System checks the availability of items.
	5. System prompts the customer for the payment type.
6. Customer chooses payment type.	
	7. If payment type is “credit card payment”, see Section Credit Card Payment. If payment type is “cheque payment”, see Section Cheque Payment.
	. . .
	23. System sends a confirmation message to the customer that the books have been shipped.

Typical Process Description

Actor Action	**System Responsibility**
Alternative 1:	
6. Customer chooses to cancel the sale.	
	7. ...
Section Credit Card Payment:	
1. Customer submits credit card number.	
	2. System sends credit card information to the Credit Card Authorization Service.
	3. System receive authorization from Credit Card Authorisation Service.
Exception 1:	
	2. System cannot connect to the Credit Card Authorization Service.

Use Case Template

- Use case name
- Version
- Goal
- Summary
- Actors
- Pre-conditions
- Triggers
- Basic course events
- Alternative paths
- Post-conditions
- Business rules
- Notes
- Author and date

http://en.wikipedia.org/wiki/Use_case
http://en.wikipedia.org/wiki/Use_case_diagram

Use Case Descriptions

Use case number

- A unique number for referencing UC in the rest of SRS
- Use cases are numbered UC1, UC2

Name

- Indicating what is captured by UC
- **UC names should start with a verb**

Authors

- Names of the people who wrote the use case

Use Case Descriptions

Event/Pre-condition

- Description of the event that initiates the use case; indicate information that is passed as input with the event
  - A use case should be triggered by a single event
  - Preconditions are noteworthy condition that are assumed to be true before beginning a scenario (not tested in the scenario)

System

- A declaration of what is considered to be the system for the use case
  - Business (interaction with business)
  - System (interaction with software)
- Business use cases describe how the business as a whole deals with customers

Use Case Descriptions

Actors

- List of the actors that participate in the use case, giving use case's initiator as the first element of the list
- The Actors' names should always be capitalized within the use case

Overview/Post-conditions

- Brief 2-3 sentence description of use case; this overview serves also as a high-level description of the use case.
- Describe what should be true on successful completion of the use case.

References

- List of the numbers of all requirements captured by the use case.

Use Case Descriptions

Related Use Cases

- List of the numbers of all related use cases; for each element of the list, describe the relationship of the identified use case to UC

Typical Process Description

- In a multi- (or single) column format, a description of the most usual instance scenario of the use case, the so-called normal interaction of actors and the system that leads to the successful outcome of the process that the use case captures. This is also called the main scenario or basic flow.
  - One column for each actor or process that is visible at the user’s level.
  - Sometimes, there will be only two columns, at least at the highest level view of the system, (1) the user or initiator of the system and (2) the system itself.

Use Case Descriptions

Typical Process Description

- In the left-most column, first row, list the initiator's actions.
- In each of the remaining rows, the reactions by one of the system's processes to the initiator's or other actor's actions are listed in the appropriate column.
- Typical actions:
  - Interaction between actor and system (input/output)
  - Validation by system
  - State change in the system (e.g., record some information)

Use Case Descriptions

Typical Process Description

- Indicate branches on certain conditions (*e.g.*, "see Section Credit Card Payment"). Branch may refer to another use case described elsewhere or subsections of this use case.
  - Branches must be based on conditions that the system or an actor can detect.
  - Alternatively, branches are not indicated in the main scenario, but later sections, show a branch of step 7 as "7a.."
- Subsections describe actions on branches.
- Subsections are assumed to merge back with the main flow, unless they indicate otherwise.

Use Case Descriptions

Alternative Flows

- Subsections for different actions that an actor can take in the main scenario. Start the line numbers at the point where the alternative flow diverges from the main scenario.

Exceptions/Extensions

- Subsection for alternative behaviours of the system based on certain conditions.

- Be careful to make it clear the scenario (main or subsection) to which alternative flows or exceptions belong.
- Almost every step can fail in some way.

Goals and Scope of a Use Case

- Focus on the question
  - How can we provide observable value to the user, or fulfill their goals?
  - Rather than thinking of the system requirements as a list of features or functions.
- Focus on the right level of abstraction
- Elementary Business Process (EBP)
  - A task performed by one person in one place at one time
  - In response to a business event
  - Adds measurable business value
  - Leaves the data in a consistent state
  - Define one EBP-level use case for each user goal
  - Name the use case after the goal, starting with a verb (e.g., "Process Sale")
  - A subtask (e.g., exception) that occurs in several base use cases can be factored out into its own use case to avoid duplication (i.e., just like OO exceptions)
  - Collapse CRUD goals (i.e., create, retrieve, update, delete goals) into a single use case, named "Manage *X*".

Elementary Business Process–EBP

- Which of these are EBPs?
  - Negotiate a supplier contract
  - Rent videos
  - Log in
  - Start up system
  - Print a document

- Can model some non-EBPs in your use case collection, but focus should be on EBPs

Elementary Business Process–EBP

- Which of these are EBPs?
  - Negotiate a supplier contract
    - Not doable by a single person in a single session
  - Rent videos
    - Yup
  - Log in
    - A system event, not a business event
    - Not very interesting
  - Start up system
    - A system event, probably trivial
  - Print a document
    - No business value

Process for Identifying Use Cases

1. Choose a system boundary
2. Identify primary actors
3. For each actor, find his or her or its goals
4. Define a use case for each goal
5. Identify the possible variations and error conditions
6. Define relationships among actors
7. Decompose complex use cases into sub-use cases
8. Organize normal alternatives as extension use cases

Common Use Case Mistakes

[These are all bad things! List adapted from Iconix]

Write function requirements instead of usage scenario text

- Requirements state “what the system under design (SUD) shall do” whereas scenarios describe actions that the user takes and expected responses of the SUD
  - Don’t model the SUD per se, model the interactions instead.

Describe attributes and methods rather than usage

- This is inappropriate attention to details; you’ll get bogged down quickly.
- Concentrate on the basic tasks and the abstract details.

Common Use Case Mistakes

Write from a non-user's perspective or using passive voice

- Use cases are all about what users expect from the system; these are the "real requirements".
- Use of the passive is to be avoided
  - Present tense active voice verb phrases are much more effective.

Describe only user interactions; ignore system responses.

- Need to detail what the system is doing (abstractly) "under the hood". This is what you are trying to discover to be able to build the systems eventually
  - For example, validate ID, prepare invoice, generate error message

Common Use Case Mistakes

Omit text for alternative courses of actions.

- Don’t “punt” on alternatives too long; these details are just as important.

Spend a month debating whether to use «`includes`» or «`extends`»

- Make a decision and live with it; it’s good to review and rethink but don’t fall victim to “analysis paralysis”.

Focus on something other than what’s “inside” a use case (e.g., what happens before or after)

- Do not spend much time on modelling pre- or post-conditions.

Use Case Limitations

- Use cases cannot express systems which do not have many externally visible behaviors and functionality
- Instead we can use other techniques like simple text sentences or data modeling techniques

Examples of Use Case Limitations

- Algorithmic and computational intensive systems (e.g., satellite tracking or optimization systems)
  - Use mathematical expressions and statistical algorithms
- Embedded systems
  - Use state machine diagrams and temporal logic expressions
- Parsers, compilers, code transformers
  - Use state machines

Other Benefits of Use Cases

- Easy to write and read relative to other requirements methods
- Forces developers to think through the perspective of users
- Help engage and interacting with the users (i.e., easy to read)
- Are useful for design and testing
- Serve as input for the user documentation (step-by-step) instructions