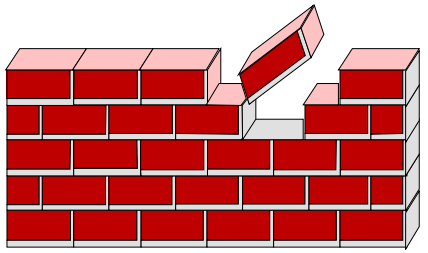
CompSci 230
Software Construction

Lecture Slides #4: Use Cases S1 2015

Agenda & Reading

- Topics:
  - Review (or learn for the first time)
    - What are the major steps in an Object-Oriented Design process?
  - Introduction to Use Case modelling
    - What? A process of determining what the stakeholders require – by decomposing their requirements into tasks (or "use cases") for each class of stakeholders.
    - How? Stakeholder Identification, Requirements Elicitation, Use Case Diagrams
    - Why learn this? Use cases are widely used in the industry, because they seem to work pretty well, they aren't very expensive to develop, and they are at a good level of detail for end-users.
    - Major alternatives (not taught in this course): user stories (for agile development), formal specifications (for safety-critical software).
- Reading:
  - D. G. Firesmith, "Use Cases: the Pros and Cons", in *The Wisdom of the Gurus*, SIGS Reference Library, 1996. Available: http://www.ksc.com/article7.htm.
- To learn more (optional reading):
  - A. Ramirez, "Requirements Capture", in *ArgoUML User Manual*, v0.34, 2011.
  - Object Management Group, "Use Cases", in *OMG Unified Modeling Language (OMG UML) Superstructure*, v 2.4.1, 6 August 2011.

Software Design (review)



- **Communication**:
  - identify stakeholders, find out what they want and need.
- **Planning**:
  - list tasks, identify risks, obtain resources, define milestones, estimate schedule.
- **Modeling:**
  - develop structure diagrams and use cases, maybe some other UML artifacts.
- **Construction**:
  - implement the software, with assured quality.
- **Deployment:**
  - deliver the software, then get feedback for possible revision.

To learn more:

R. Pressman, *Software Engineering: A Practitioner's Approach*, 7th Ed., 2010, pp. 14-15.

Stakeholder Identification

- Identify a variety of stakeholders, by asking yourself:
  - Who is likely to be affected by, or to have an effect on, this system?
- Classify the stakeholders you know about.
  - Anyone who will directly use the system is a stakeholder.
  - Anyone who will be indirectly affected (in a major way) is a stakeholder.
    - Anyone who pays for, or otherwise controls. the design of the system is a stakeholder.
    - Advertisers are important stakeholders for Google's online search service.
  - Governments are stakeholders, if their laws constrain the design of a system (e.g. because citizens could be greatly harmed by the system).
  - Note: use cases depict the requirements of direct stakeholders (users), but you'll have to use another method (e.g. natural language) to describe the requirements of indirect and external stakeholders.
- Reflect on your classification – have you missed an important class?

Use Case Analysis

- To start developing use cases, ask yourself:
  - What useful tasks could be performed by my system, upon request by a user?
  - You probably won't "get it right" at first. (It'll never be perfect, but could be improved…)
- To validate your current set of use cases, talk to stakeholders!
  - Ask them "Would you use a system, if it would help you do …?"
  - If they start telling you how they want the system to handle a use-case, then you have validated this use-case.
    - You should record their detailed requirements, in natural language, as notes which accompany your use case.
    - If their detailed requirements are infeasible or contradictory, you should take careful note of this!
  - If they tell you about some other task they'd like the system to help them with, you should document this as a possible use-case.
    - Your system can't do everything!
    - Whenever you discover that you can't deliver on all use cases within your current resources, you should communicate with your stakeholders to negotiate a feasible set.

An Example: Video System

- John's Video Store is an Information System which supports the following business functions:
  - Recording information about videos the store owns
    - This database is searchable by staff and all customers
  - Information about which customer is renting which videos
    - Access by staff, and also by customers who is asking about themselves.
  - Staff are able to record video rentals and returns by customers.
    - John doesn't trust his customers to make these entries in their own records!
  - Staff can maintain customer, video and staff information.
    - Privacy requirements: customers cannot access information about other customers, personal information about customers must be accurate and relevant to John's Video Store, …
  - Managers of the store can generate various reports.

Who are the stakeholders?

- John's Video Store is an Information System which supports the following business functions:
  - Recording information about videos the store owns
    - This database is searchable by staff and all customers
  - Information about which customer is renting which videos
    - Access by staff, and also by customers who is asking about themselves.
  - Staff are able to record video rentals and returns by customers.
    - John doesn't trust his customers to make these entries in their own records!
  - Staff can maintain customer, video and staff information.
    - Privacy requirements: customers cannot access information about other customers, personal information about customers must be accurate and relevant to John's Video Store, …
  - Managers of the store can generate various reports.

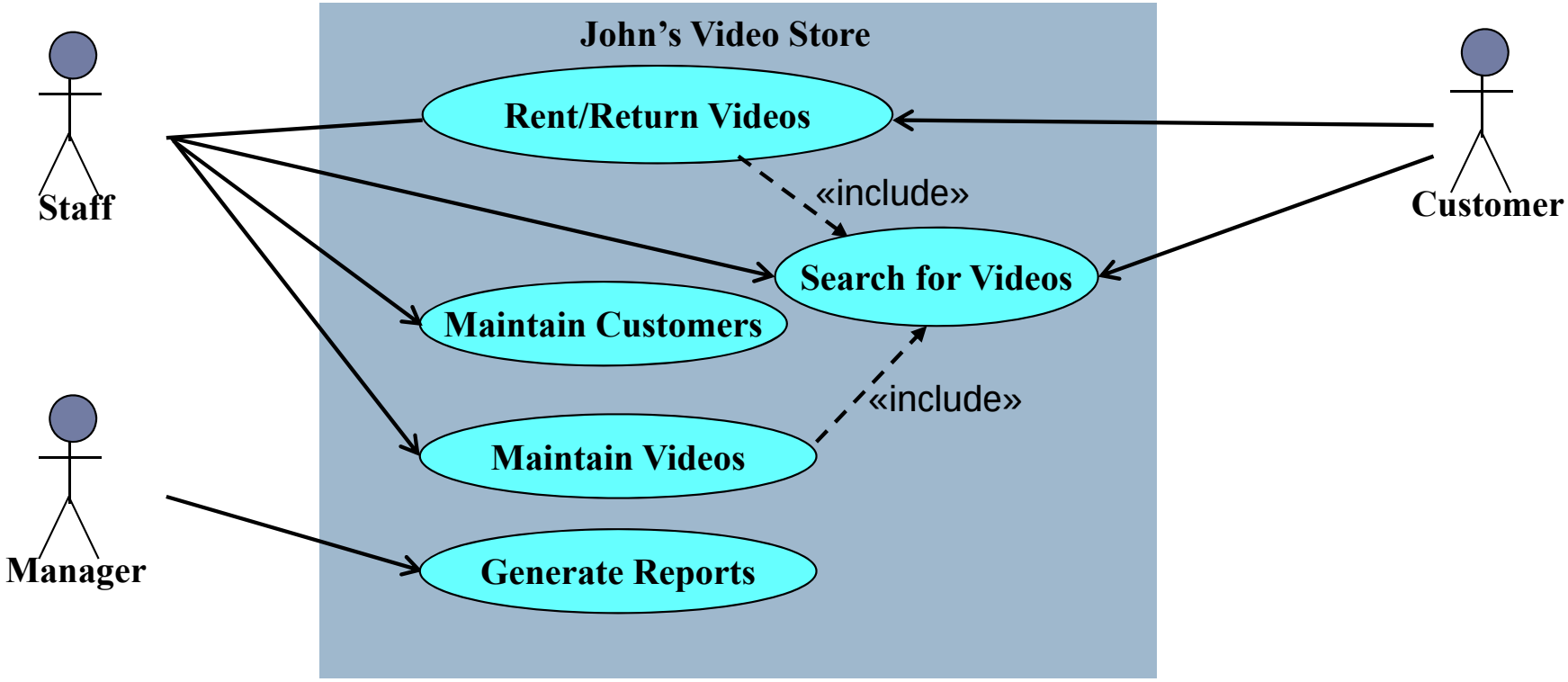
What are the tasks?

- John's Video Store is an Information System which supports the following business functions:
  - Recording information about videos the store owns
    - This database is searchable by staff and all customers
  - Information about which customer is renting which videos
    - Access by staff, and also by customers who is asking about themselves.
  - Staff are able to record video rentals and returns by customers.
    - John doesn't trust his customers to make these entries in their own records!
  - Staff can maintain customer, video and staff information.
    - Privacy requirements: customers cannot access information about other customers, personal information about customers must be accurate and relevant to John's Video Store, …
  - Managers of the store can generate various reports.

Requirements Documentation

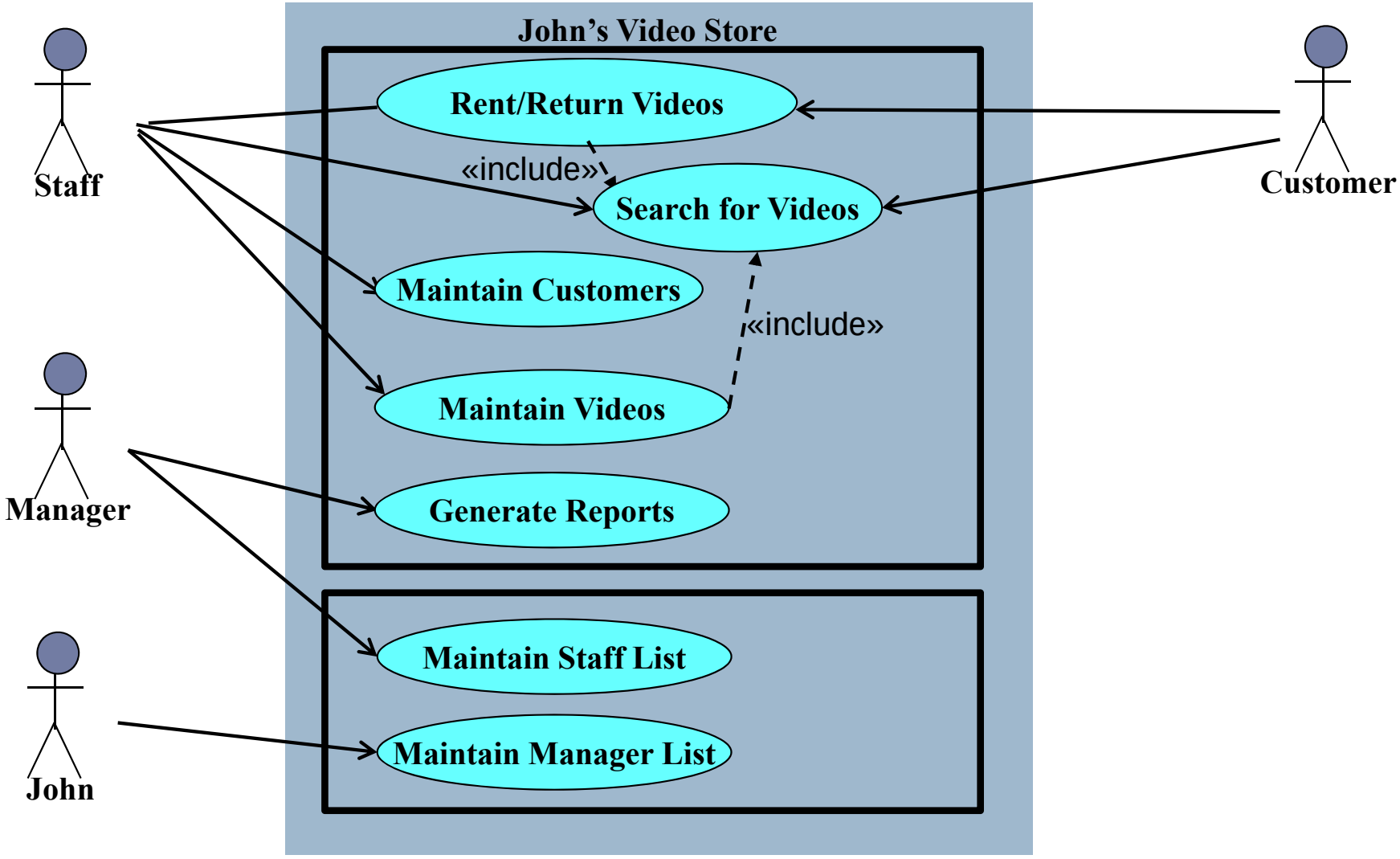
- Use case descriptions
  - A brief statement of what happens during each use case.
  - The previous slide is a good start on this, but it's not well-organised.
- Use case diagrams show
  - **Stick-figure** actors, interacting with the system (a **box**).
    - Choose easily-understood names for your classes of stakeholders!
    - John's Video Store might have three actors: Customer, Staff, and Manager.
    - (Hmmm… is John an actor? Does he have a special use-case which is so important that we must add it to our diagram? Hold this question…)
  - **Ovals** ("use cases") within the box, with easily-understood names, e.g. "Rent a video".
  - **Lines** ("associations") between actors and ovals.
  - Optionally: arrowheads, extension cases, included cases, subsystems.

Example: John's Video Store

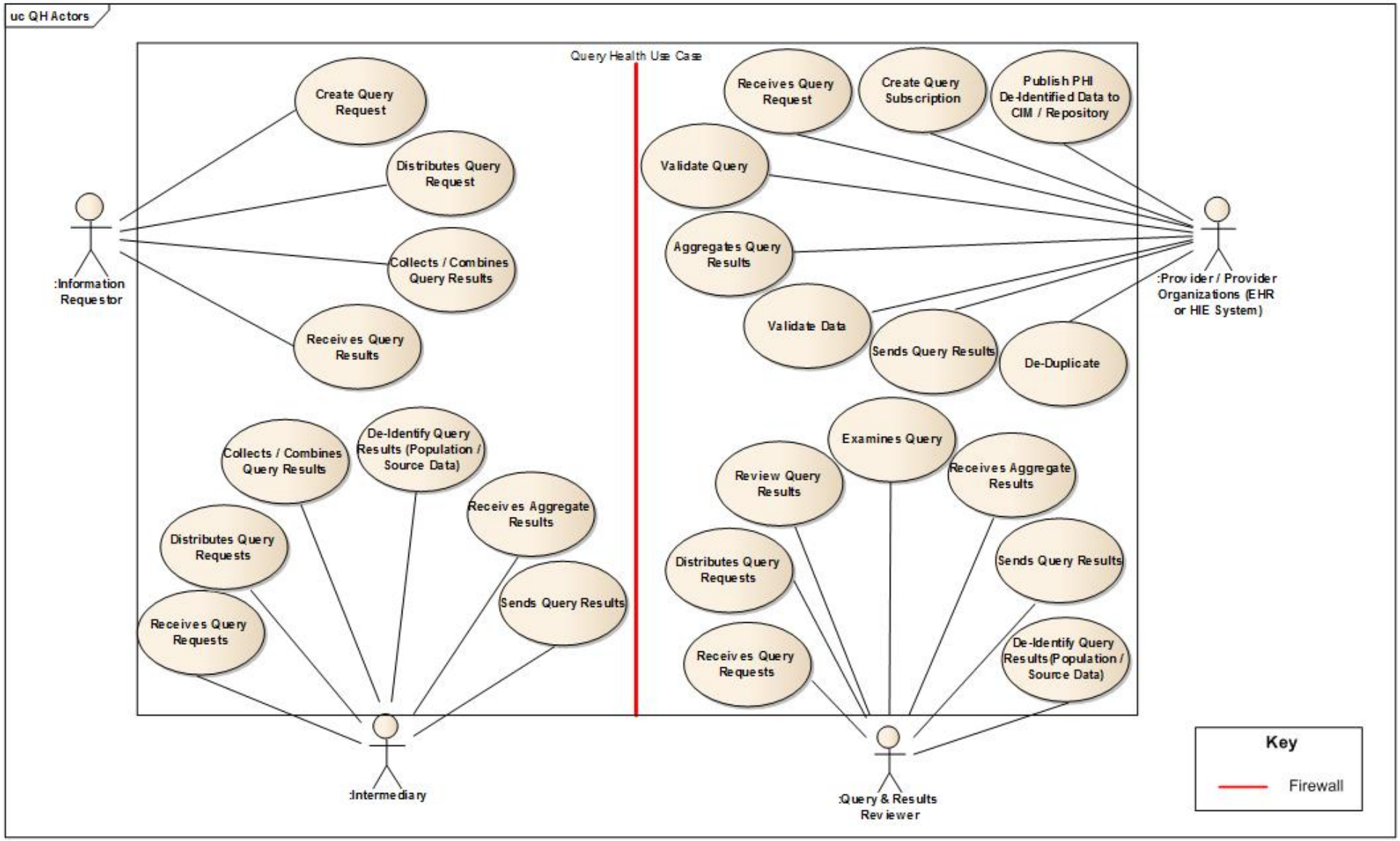


- "Include is a DirectedRelationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. …
- "An include relationship between use cases is shown by a dashed arrow with an open arrowhead from the base use case to the included use case. The arrow is labeled with the keyword «include»." [OMG UML v2.4.1, §16.3.5]

John's Video Store, with HR module



Example: Query Health Use Case



Finalized: 2011-11-16

http://wiki.siframework.org/Query+Health+-+Consensus+Approved+Use+Case

Video System – Designing the Classes

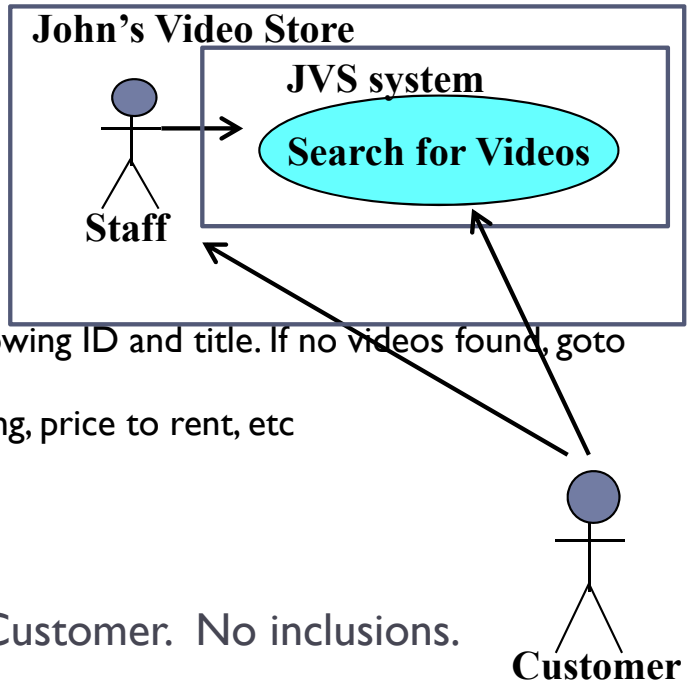
- In this system, information stored includes:
  - Videos - unique ID; title; category (children's, drama, comedy, etc); cost per night to rent; number of copies video store has available; rating
  - Staff - unique ID; name; password; position
  - Customers - unique ID; name; password; address; phone #
  - Rentals - date rented, customer who rent the video and whether video returned
- Functions this system provides include:
  - Staff can add, update, delete and find videos
  - Staff can add, update, delete and find people.
  - Staff can rent out videos to customer and indicate videos have been returned.
  - Various reporting functions e.g. number of videos rented this month are provided for managers.

Use Case Descriptions

- Use case descriptions should be detailed enough that system analysts can
  - design the classes (by grouping attributes and decomposing functions), and
  - determine the non-functional requirements:
    - "what the system should be" (or always be doing), as distinguished from "what the system should do, upon request";
    - "what the system shouldn't do" (security requirements);
    - usability, auditability, performance, efficiency, capacity, scalability, extensibility, availability, reliability, integrity, recovery, compatibility, portability, maintainability, transparency, legal conformance, …

Semi-formal Use Cases

- In some development environments (e.g. in the IBM Rational Unified Process), use cases are semi-formal documents with a required structure e.g.
  - Title: the "goal the use case is trying to satisfy" [Fowler, 2004]
  - Main Success Scenario: a numbered list of steps
  - Step: "a simple statement of the interaction between the actor and a system" [Fowler, 2004]
  - Extensions: separately numbered lists, one per extension
  - To learn more, see the Wikipedia article on "Use Case". (But don't worry about extensions. The focus in CompSci 230 is on the basics!)
- Example: a semi-formal use case for SearchForVideos
  - 1. Used by Staff via an application to query for videos by title.
  - 2. Event Flow:
    - 2.1 Repeat Until Exit Program
      - 2.1.1 Staff types in part of title in text field,
      - 2.1.2 Staff clicks "Search" button and a list of matching videos are returned showing ID and title. If no videos found, goto step 2.2. If error, goto step 2.3.
      - 2.1.3. Staff types in a ID. More information is displayed about the video e.g. rating, price to rent, etc
      - 2.1.4 Exit Program
    - 2.2 No videos found - error message displayed. Goto 2.1.1
    - 2.3 Database Error – error message displayed. Goto 2.1.1
  - 3. Related Actors and Use Cases: Staff may perform this search for a Customer. No inclusions. Included in Rent/Return Videos and Maintain Videos.
  - 4. Special conditions: NONE



How to draw UML class diagrams?

- Sketch by hand
- Use a general-purpose graphics editor
- Use ArgoUML, or some other specialised graphics editor
- Ideally, your UML tool is integrated with your IDE.
  - Forward engineering: document your requirements with use cases, develop your design with class diagrams, then start coding.
  - Reverse engineering: inspect the code to discover its class structure and use cases.
    - ArgoUML does a good job of reverse-engineering class diagrams.
    - ArgoUML is clueless about reverse-engineering use cases. (Do you understand why this form of reverse-engineering is very difficult?)

Alternatives to use cases

- **Story**, in agile development:
  - a one-sentence description of a feature which could be implemented quickly (i.e. tomorrow, or by the end of this week).

"As a member of John's staff, I want to search for my customers by their first name, last name, or by their first and last name."

- **Formal specification**, in safety-critical development:
  - a precise statement, in a formal language, of
    - the post-conditions which will hold after a system action is completed,
    - given some pre-conditions (which are also formally specified),
  - with some accompanying, explicit, and validated assumptions about the system and its environment.

$$
\begin{array}{|l}
\hline
FindBirthday \\
\hline
\Xi BirthdayBook \\
name? : NAME \\
date! : DATE \\
\hline
name? \in known \\
date! = birthday(name?) \\
\hline
\end{array}
$$

Review

- Use cases are functional descriptions of what the system should do for its users.
  - Use case diagrams depict Actors, the system, and the tasks performed by the system that are important to the Actors.
  - If use case descriptions are sufficiently detailed, then they are very helpful in OO design.
  - Use case diagrams are orthogonal to OO design, except in their identification of Actors (= classes of users).
- Use cases are commonly used in commercial software development, but there are some important alternatives.
- Learning goals for this unit:
  - If you're aiming for an A in this class, you should be able to discuss the strengths & weaknesses of use case analysis as a methodology for requirements capture.
  - If you're aiming for a B or better, you should be able to do a good job of drawing up a set of use cases from an informal description.
  - If you're aiming for a C or better, you should be able to do a good job of interpreting the information presented in a use-case diagram or description. (Practice in Quiz 1.)