Use Cases

- A use case characterizes a way of using a system
  - It represents a dialog between a user and the system, from the user's point of view
    - It captures *functional* requirements

Benefits of doing use cases

- Establish an understanding between the customer and the system developers of the requirements (success scenarios)

- Alert developers of problematic situations (extension scenarios)
- Capture a level of functionality to plan around (list of goals)

Terminology

Actor: someone who interacts with the system

Primary actor: person who initiates the action

Goal: desired outcome of the primary actor

Level:

- summary goals
- user goals
- subfunctions

Do use cases capture these?

- Which of these requirements should be represented directly in a use case?

1. Order cost = order item costs * 1.06 tax.
2. Promotions may not run longer than 6 months.
3. Customers only become Preferred after 1 year
4. A customer has one and only one sales contact
5. Response time is less than 2 seconds
6. Uptime requirement is 99.8%
7. Number of simultaneous users will be 200 max

Styles of use cases

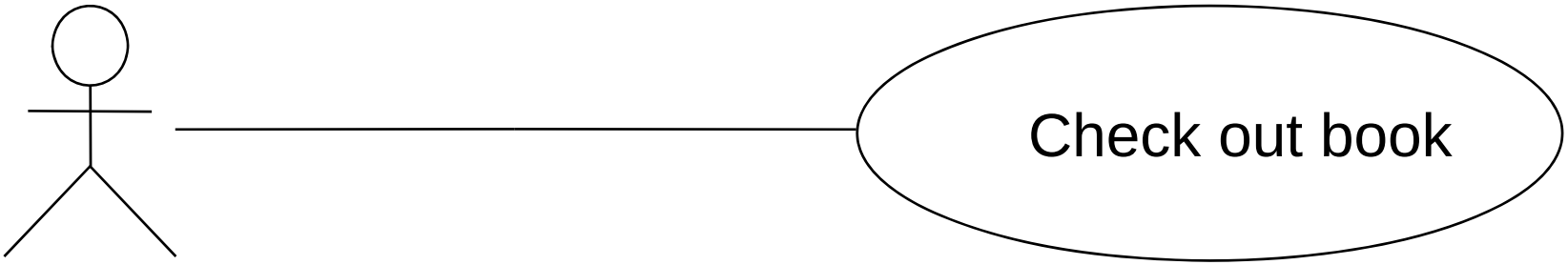
1. Use case diagram (UML/Visio/Violet)
   - shows all use cases in system
2. Informal use case
3. Formal use case

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

1. Use case summary diagrams

The overall list of your system's use cases can be drawn as high-level diagrams, with:

- actors as stick-men, with their names (nouns)
- use cases as ellipses with their names (verbs)
- line associations, 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



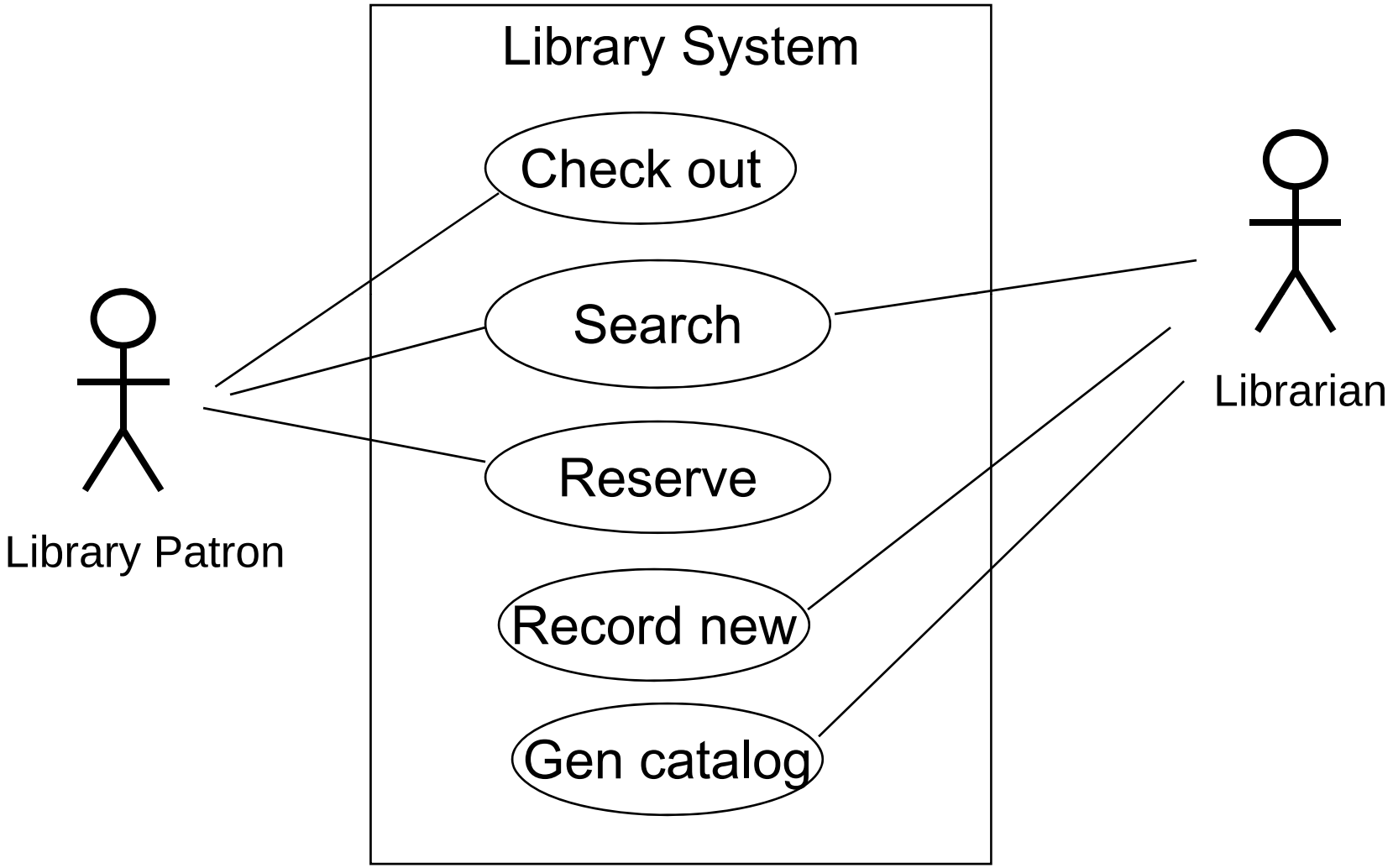
Library patron

Use case summary diagrams

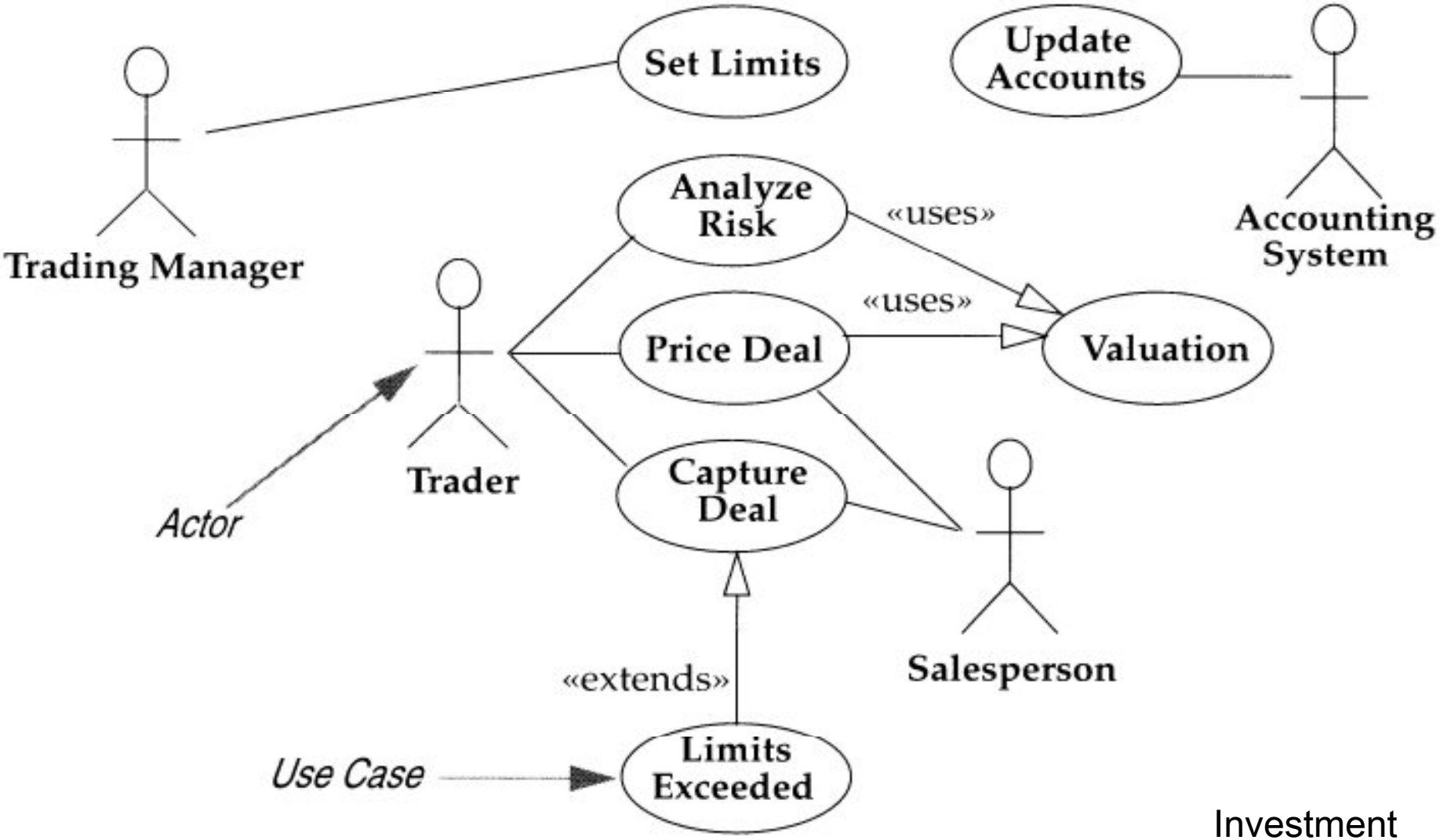
It can be useful to create a list or table of primary actors and their "goals" (use cases they start). The diagram will then capture this material.

Actor	Goal
Library Patron	Search for a book
	Check out a book
	Return a book
Librarian	Search for a book
	Check availability
	Request a book from another library

Use case summary diagram 1



Use case summary diagram 2



Investment System

UML/Visio

- Quick demo

2. Informal use case

Informal use case is written as a paragraph describing the scenario/interaction

- Example:
  - Patron Loses a Book
    The library patron reports to the librarian that she has lost a book. The librarian prints out the library record and asks patron to speak with the head librarian, who will arrange for the patron to pay a fee. The system will be updated to reflect lost book, and patron's record is updated as well. The head librarian may authorize purchase of a replacement tape.

3. Formal use case

Goal	**Patron wishes to reserve a book using the online catalog**
Primary actor	**Patron**
Scope	**Library system**
Level	**User**
Precondition	**Patron is at the login screen**
Success end condition	**Book is reserved**
Failure end condition	**Book is not reserved**
Trigger	**Patron logs into system**

Main Success Scenario	1. Patron enters account and password 2. System verifies and logs patron in 3. System presents catalog with search screen 4. Patron enters book title 5. System finds match and presents location choices to patron 6. Patron selects location and reserves book 7. System confirms reservation and re-presents catalog
Extensions (error scenarios)	2a. Password is incorrect 2a.1 System returns patron to login screen 2a.2 Patron backs out or tries again 5a. System cannot find book 5a.1 …
Variations (alternative scenarios)	4. Patron enters author or subject

Steps in creating a use case

1. Identify actors and their goals

   What computers, subsystems and people will drive our system? (actors)

   What does each actor need our system to do? (goals)

Identify actors/goals exercise

- Let's identify some major actors and their goals for your projects
  - U-Mail
  - Notepad
  - Visual Registration
  - OfCourse
  - Facebook
  - Foresee

2. Write the success scenario

- Main success scenario is the preferred "happy path”
  - 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
- Note the failure condition separately, after the main success scenario
- Label with step number and letter:
  - 5a failure condition
    - 5a.1 use case continued with failure scenario
    - 5a.2 continued

- Exercise: What happens if a student looks up a course in OfCourse, and it doesn't exist?

4. List the variations

- Many steps can have alternative behaviors or scenarios
- Label with step number and alternative
  - 5'. Alternative 1 for step 5
  - 5''. Alternative 2 for step 5

Exercise: What are some variations that arise in arranging a carpool with Foresee?

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
- A good use case summary fits on a page

Exercise: Project use case

- Each project team break into 2 groups
- Each group write a use case for your product
- Choose one per project team to share with the class

Your SRS doc can use these use cases!

Pulling it all together

How much is enough?

You have to find a balance

- comprehensible vs. detailed correctness
- 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

Words of Wisdom 5

After you create a specification, go over it to:

- Eliminate all requirements not absolutely necessary
- Simplify those that are more complicated than necessary
- Substitute cheaper options when available
- Move non essentials to future releases

Words of Wisdom 5’

Agile Principle – Simplicity is Essential