CSE 403

Requirements

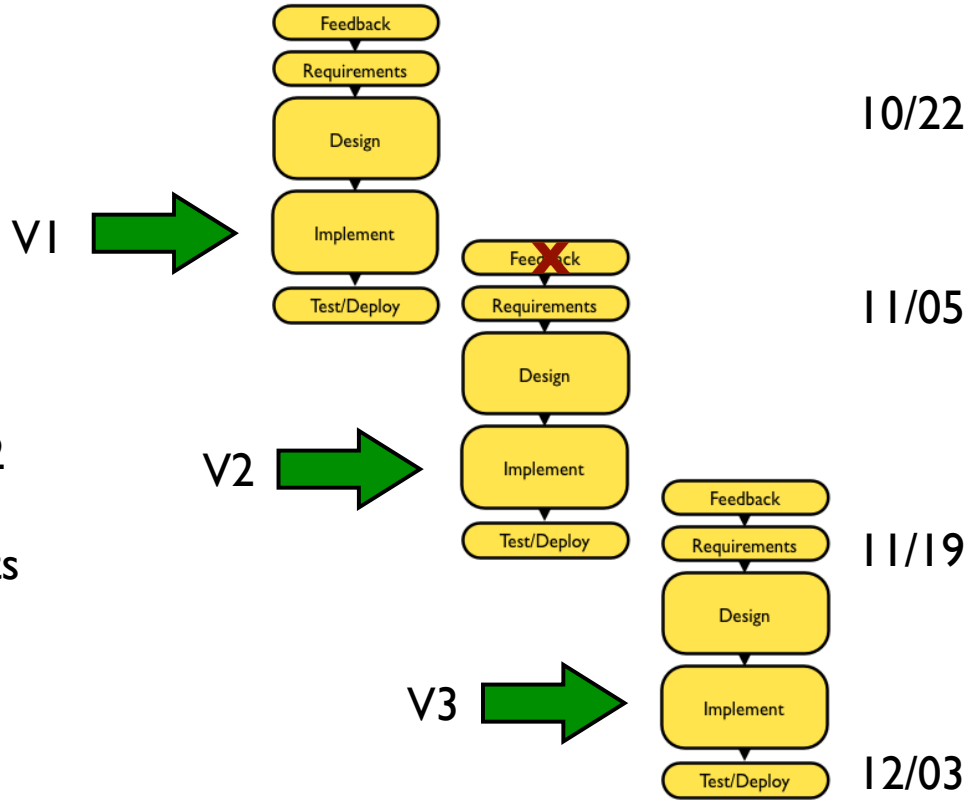
“There are only two hard problems in software engineering”
(tip of the day)

- Naming
- Cache invalidation/coherency
- Off-by-one errors

Class announcements

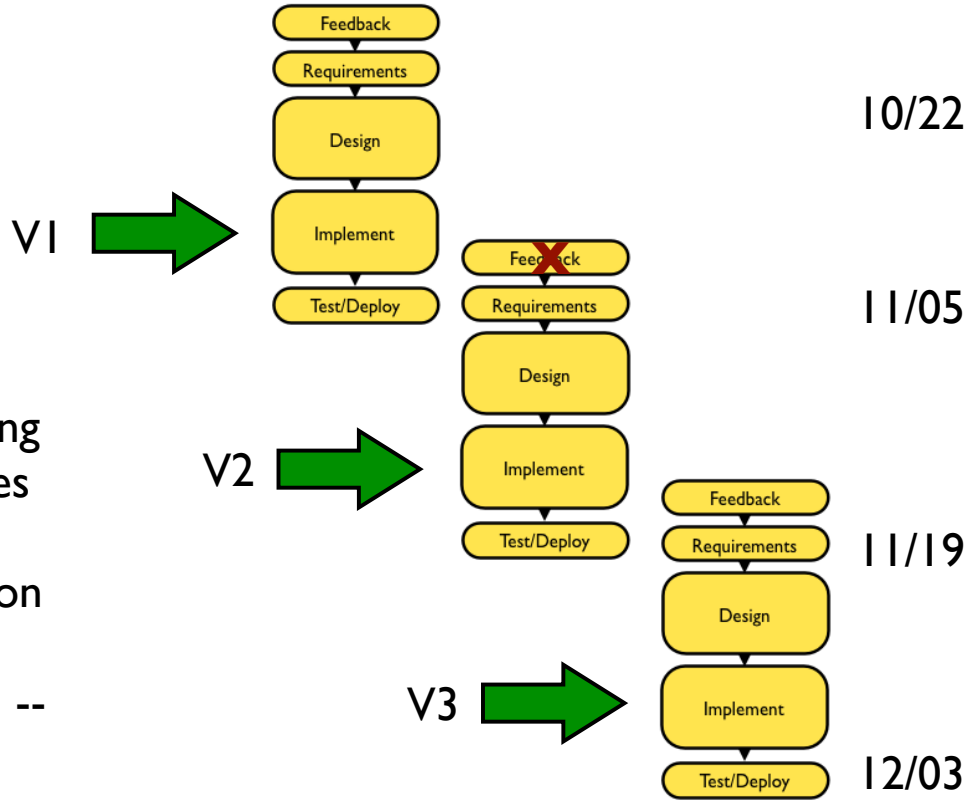
- Requirements due at 11:59PM on Monday
- Project managers must turn in weekly status report by 11:59PM on Sunday
- Your team roles should be settling down
- Hacking, experimenting, prototyping as you execute the formal assignments
- Next week: architecture, design, tools, process docs due

Parallelizing/pipelining releases: should you do it?



- Release cycle is 3 weeks w/ ~2 weeks of development
- Overlap feedback/requirements of next release with implementation of previous release
- 3 releases in 6 weeks

Parallelizing/pipelining releases: should you do it?



- Decomposing the cycle into stages is key -- as for all pipelining
- Probably need to partition roles between people doing requirements and implementation
- Doing "double" the work
- Managing the overlap is critical -- how much, where
- What are the cross cycle dependencies?

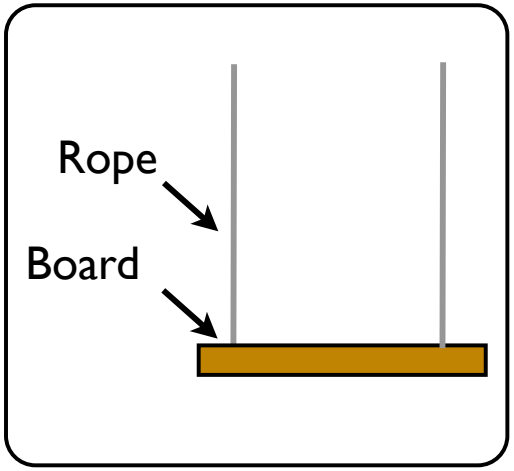
Requirements

What are you going to build?

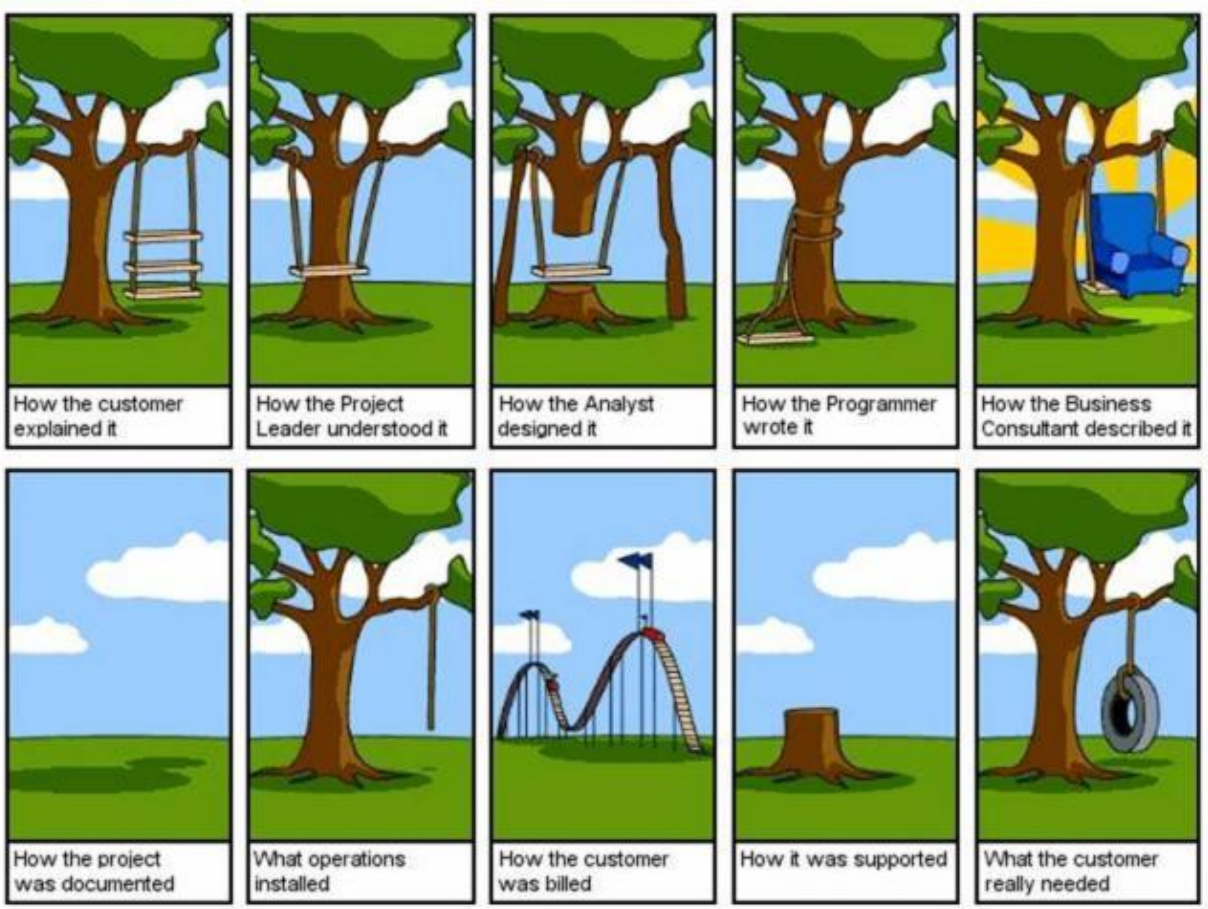
- Tell "what" not "how"
- Provide vision to show "why"
- Describe "who"
- Explain business drivers

Example: A swing

Build a swing for a tree.



Interpretation



Writing requirements is like describing an elephant

Specificity and clarity are keys

- Two ropes attached to a board
- What kind of board? What kind of rope?
- How long of a rope? How long of a board?
- What will this be used for? Who will use it?
- How much weight must it support?
- How will it get installed by the customer?
- What are the end deliverables?

Ambiguity is an enemy

- **Must I carry a dog?**
- **What about the new shoes in my shopping bag?**
- **Do dogs have to wear shoes?**
- **What about an amputee? A single shoe? A double amputee?**
- **What are shoes?**
- **What are dogs?**

Shoes Must Be Worn

Dogs Must Be Carried

Conclusion:

Specifying requirements is hard!

Why specify requirements?

- Formulate, understand, and communicate what is being built
- Set priorities on when things should be developed and deployed
- Ensure that product is aligned with overall company vision and goals

Outline

- What are requirements?
- How do we gather requirements?
- How do we specify requirements?

What are requirements?

What are requirements?

- Vision and background
- Users and use cases (role specifications)
- Functionality/Features (typically the most important)
- Scale issues
- Strategic planning
  - Spans multiple { releases, months, years, projects }
  - Road map (maybe with some date horizons)
  - Platforms to support (sometimes)

General classes of requirements

- Feature sets
- GUI
- Performance
- Reliability
- Expansibility

Good or bad requirements? (why?)

- The system will enforce a 6.5% sales tax on Washington purchases
- The system shall display the elapsed time for the car to make one circuit around the track within 5 seconds in hh:mm:ss format
- The product will never crash. It will also be secure against hacks
- The server backend will be written using PHP or Ruby on Rails
- The system will support a large number of connections at once, and each user will not experience slowness or lag
- The user an choose a document type from the drop-down list

Who specifies the requirements?

- Mostly the product manager (with input from lots of others)
- Engineering team (lead) -- usually functional requirements, and refinements
- CEO (fly-by-requests -- be careful!)
- Typically it's interaction between the product manager and the engineering team -- that interaction is often confrontational and painful!
- Skunk works or "bottom up" requirements (engineers)

How do we gather requirements?

Many ways to get requirements

- Intuition/Domain knowledge
- By committee
- CEO says!
- Iterative process between management and engineering (refinement)
- Customer
  - ews, focus groups, test/measure
  - dies
- nts!

Many times customers don't know they want something that doesn't exist!

“We” might not know either

“There is no reason for a person to have a computer in their home.”

-- Ken Olson, CEO and founder of Digital Equipment Corporation, 1977

Users and use cases

Users and use cases

- Actors -- users of the system
- Use cases -- examples on how the actor will use the system (including involvement of other actors)
- Goal: Desired outcome of the actor in the use case (success case)
- Detailed sequencing on typical/success case
- Some expression of exceptional conditions

Use case

- Example scenario on how the system is used by a user
- Expression from a user's point of view
- It implicitly shows features/functional requirements
- But can't capture all features or exceptions
- Example: Jane has a meeting at 10AM; when Jim tries to scedule another meeting for her at 10AM, he is notified about the conflict

Qualities of a good use case

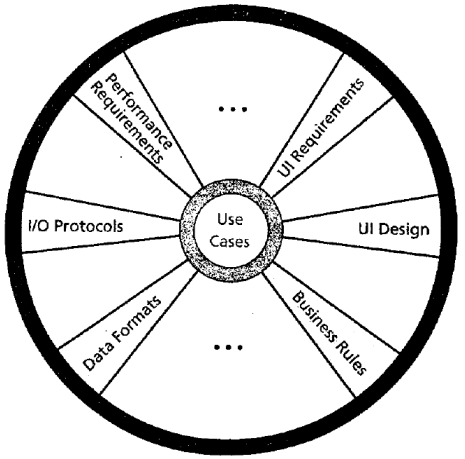
- Starts with a request from an actor
- Ends with the product of all the answers to the request
- Defines the interactions between the system actors related to the function
- View point is from the perspective of the actor
- Doesn't describe the GUI in detail
- Has 3-9 steps in the main success scenario
- Is easy to read
- Summary fits on a page

Cockburn's template for use cases

Alistair Cockburn's suggested outline for functional requirements:

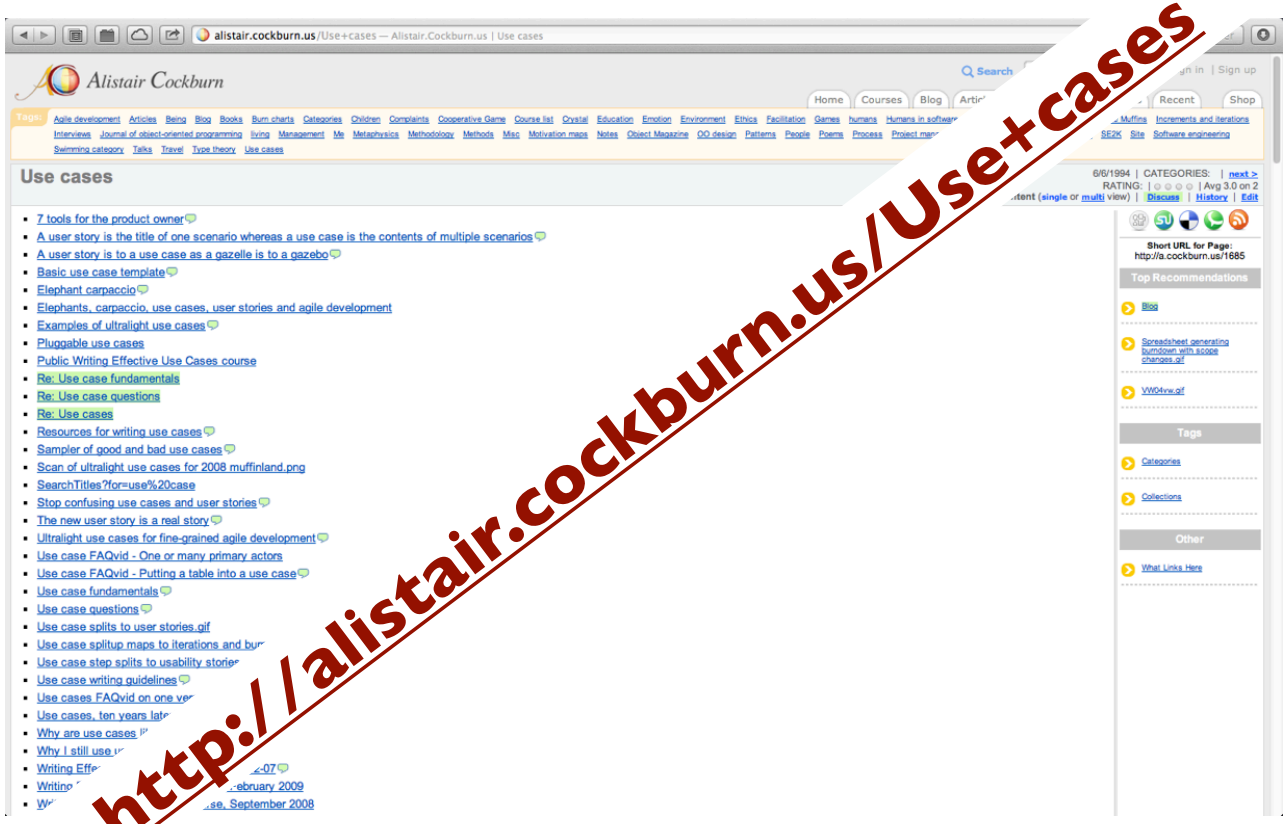
1. purpose and scope
2. terms / glossary
3. **use cases**
4. technology used
5. other
   a. development participants, visibility, ...eap?), ...dencies
   b. business ...aints
   c. performa... ...mands
   d. security (now a hot topic), documentation
   e. usability
   f. portability
   g. unresolved / deferred
6. human issues: legal, political, organizational, training

Is this useful?



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

Alistair Cockburn's "Writing Effective Use Cases"



Practically speaking...

- Actors
- Preconditions
- Triggers (what starts the use case)
- Minimal/success guarantees (end condition)
- List of steps to a successful scenario
- Failure end conditions
- Extensions, alternative paths

Example

Use Case 1 **Buy Stocks over the Web**

Primary Actor: Purchaser
Scope: Personal Advisors / Finance package (PAF)
Level: User goal
Stakeholders and Interests:
Purchaser—wants to buy stocks and
automatically.
Stock agency—wants full pur
Precondition: User already h
Minimal Guarantee: Suff
that something went
Success Guarante
the user's port
Main Succ
1. Pur
2.

at PAF can detect
s.
e purchase; the logs and

de, Schwab, etc.) from user.
e, retaining control.
k from the web site.
n the web site and updates the purchaser's portfolio.
new portfolio standing.

s a web site PAF does not support:
gets new suggestion from purchaser, with option to cancel use case.
ilure of any sort during setup:
System reports failure to purchaser with advice, backs up to previous step.
3a2. Purchaser either backs out of this use case or tries again.
4a. Computer crashes or is switched off during purchase transaction:
4a1. (What do we do here?)
4b. Web site does not acknowledge purchase, but puts it on delay:
4b1. PAF logs the delay, sets a timer to ask the purchaser about the outcome.
5a. Web site does not return the needed information from the purchase:
5a1. PAF logs the lack of information, has the purchaser update questioned purchase.

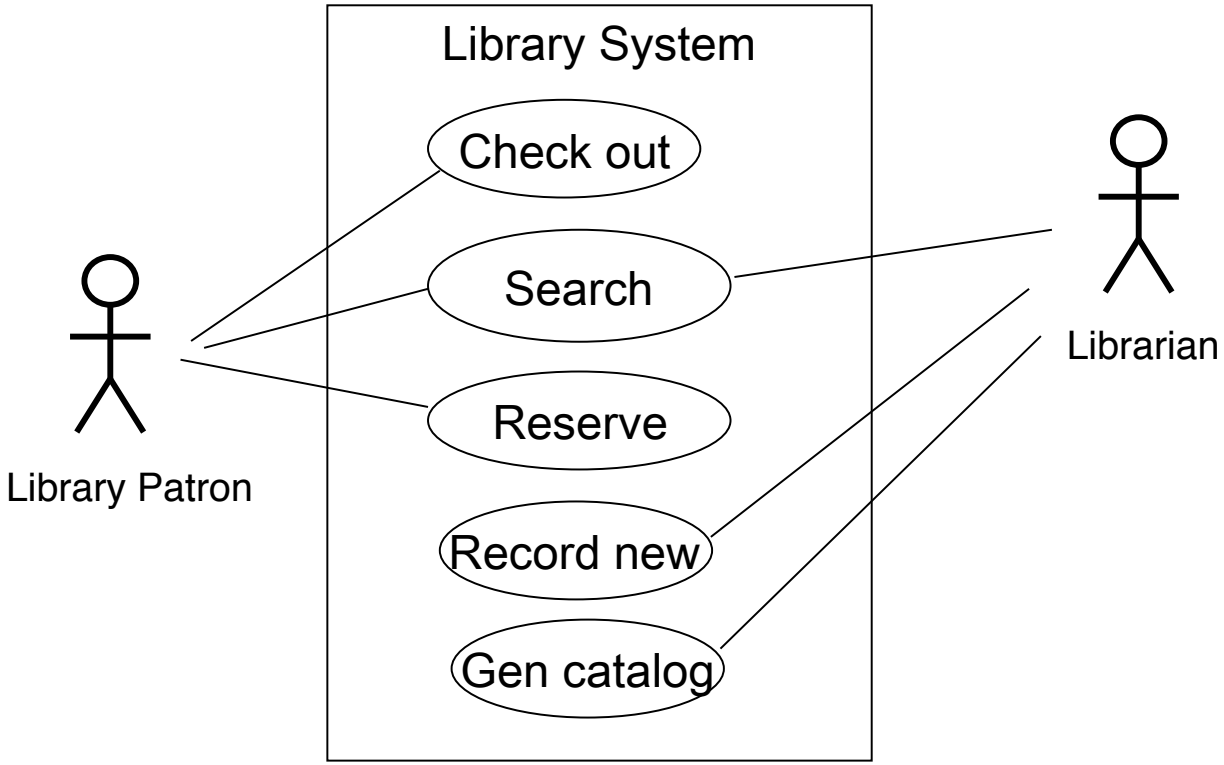
But this is pretty verbose and time consuming...

Simplify....

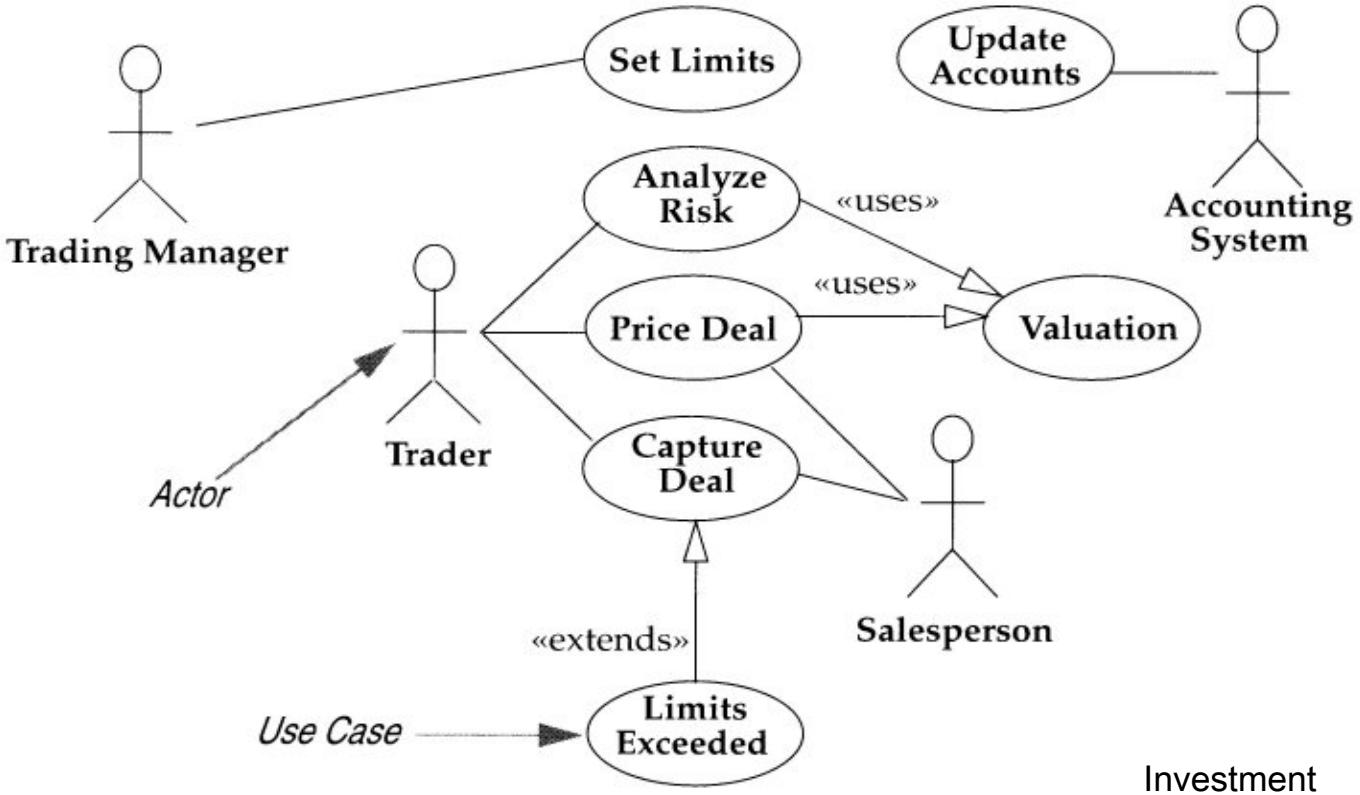
- Enumerate actors
- Enumerate use cases for each actor

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

Pictures sometimes help



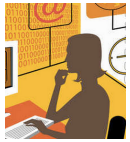
Pictures for complicated scenarios



Investment System

Example

Actors and use cases



Visitors: Find nonprofits, browse nonprofit data, leave comments, update nonprofit data, make a donation

Donors: Register/login, make donations, get receipts, see donations, crowd source fund raising campaigns, broadcast activities onto social media channels, leave comments, update nonprofit data

Nonprofits: Claim/setup nonprofit listing page, login, update profiles, see donors and donations, specify how they receive donations, write blog posts, add photo stream, manage donor relationships

Administrators (Charity Blossom employees): Manage listings, send donations, interact with visitors and nonprofits (customer support),

Example use case: Jane makes a donation

- **Actor**: Jane, a visitor; **Scenario:** Visitor makes a donation; **Precondition**: NA; **Trigger:** Jan wants to make a donation; **Success condition**: Donation made; **Failure condition**: Donation not made
- **Scenario**
  - Jane browses or searches for a nonprofit to make the donation
  - She reviews information on the nonprofit as well as the terms of service at Charity Blossom
  - She specifies the amount and payment method for the donation and commits to making the donation
  - She is presented with a receipt for the donation
- **Exceptions:** Jane can't find the nonprofit, Jane is a registered user, Jane's credit card fails to get approved, nonprofit can't take donations

Library example

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**

Library example (continued)

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

Hints

- Detailed text descriptions work pretty well
- Hard to capture all the exceptions and paths even in this simple example
- Pictures can help
- Doesn't capture all the feature/functional requirements even though the use case provides insight
- Not enough time to write out all the use cases(?) -- marginal return on time?
- Formal tools?

Feature/functional specifications

Functionality/Features

- Usually bundled into the next release cycle
- Broken down into features
- Cognizant of time frame and resources
- Harder in the beginning? Because you are making decisions that have long term implications

How to formulate functional requirements

- Divide and conquer from vision
- Listen to feedback
- Brainstorm
- mutually exclusive, completely exhaustive (?)

Examples

- Display listing information for every nonprofit (approximately 1.5MM)
- Periodically get (new) nonprofit listing information from irs.gov
- Allow visitors to leave reviews, update listing information
- Allow nonprofits to "claim" listing page by registering with CB.
- Allow nonprofit to update listing information
- Allow visitors to make a donation to help a specific nonprofit
- Require nonprofits to register/login when claiming listing

Drill down: Login and registration

- A nonprofit must be able to register and login to use the system