Conceptual Design

6.170 Recitation 8
Concepts - The Standard Template

- Name
- Purpose
- Structure (Data Model)
- Specifications (Actions + Effects)
- Operational Principle
- Misfit
Example: Upvote

Graffiti from Scotland

A beautiful landscape (unknown location).

10,328,341 views

345,469 likes
13,423 dislikes
Step 1: Define Purpose

What is the purpose of an upvote?
Step 1: Define Purpose

Measures the quality (or popularity) of content

Engages a user in the platform
Step 2: Draw Structure

Create a **data model** that includes the upvote concept
Step 2: Draw Structure

Textual Constraints: user.poster != user.upvoter
Step 3: List Specifications

What are the specifications (actions + effects) of an upvote?
Step 3: List specifications

User Actions

- upvote(post: Post, user: User)
  
  \textbf{requires}: post.poster \neq post.user \textbf{and} user \textbf{not in} post.upvoters

  \textbf{effects}: post.upvoters := post.upvoters + user

System Actions

- order(posts: Post[])
  
  \textbf{effects}: \textbf{sort} posts \textbf{by} |post.upvoters|
Step 4: State Operational Principles

What are the operational principles of an upvote?
Step 4: State Operational Principles

A registered user $u$ can make a post $p$, while another user $u^*$ can upvote $p$ incrementing its upvote count.

When posts are displayed, they are shown in order from most to least votes.
Exercise: Droplist

Step 1: Define purpose

What is the purpose of a droplist?
Step 1: Define Purpose

Help people save money.
Step 2: Draw Structure

Create a data model that includes the droplist concept
Step 2: Draw Structure
Step 3: List Specifications

What are the specifications (actions + effects) of a droplist?
Step 3: List specifications - User Actions

- **add** (droplist: Droplist, item: Item, threshold: Float, expiration: Date)
  
  **requires**: there does not exist a watch on item in droplist
  
  **effects**: add a watch to droplist which watches item for a price below threshold until expiration

- **updateThreshold** (watch: Watch, newThresholdPrice: Price)
  
  **effects**: watch is now for a price below newThresholdPrice

- **updateExpiration** (watch: Watch, newExpiration: Date)
  
  **effects**: watch now expires on newExpiration

- **remove** (watch: Watch)
  
  **effects**: delete watch and all of its relations
Step 3: List specifications - System Actions

- **notify(watch: Watch)**
  
  **requires:** the item the watch watches costs less than the watch's threshold price
  
  **effects:** sends an event about *watch*

- **removeExpired(watch: Watch)**
  
  **requires:** the item the *watch* watches expires on a date before the current date
  
  **effects:** delete *watch* and all of its relations

- **update(item: Item, price: Price)**
  
  **effects:** the item now costs *price*
Step 4: State Operational Principles

What are the operational principles of a droplist?
A user can **add** watches to a *droplist*. A user can **edit** the threshold *price* and expiration *date* of a *watch*. They can also **remove** a *watch* from its *droplist*.

The *droplist* **notifies** the user when one of the *items* it watches drops below a set *price*. The prices of the items watched in the *droplist* are **updated** to reflect the most current prices of the items in the store.