Fundamentals of Interaction Design

CSCI 3002 Fall 2018

Today

- Reflections on paper prototyping
- Usability fundamentals

Paper prototyping

- Was paper worth it?
- Reflections on the process?
- Using and citing materials
- You need to test WAY MORE than you think
- Learning to observe and understand errors is key – challenge yourself to find problems

10,000 ft view

- **Previously:** fundamentals of constructing user interfaces
  - But… they are probably bad user interfaces 😕
- **Now:** principles of user interface design
  - common wisdom about designing effective interactions
  - *background:* why is it this way?

can prototype and test user interfaces

can create interfaces that follow
best practices; explain why they work

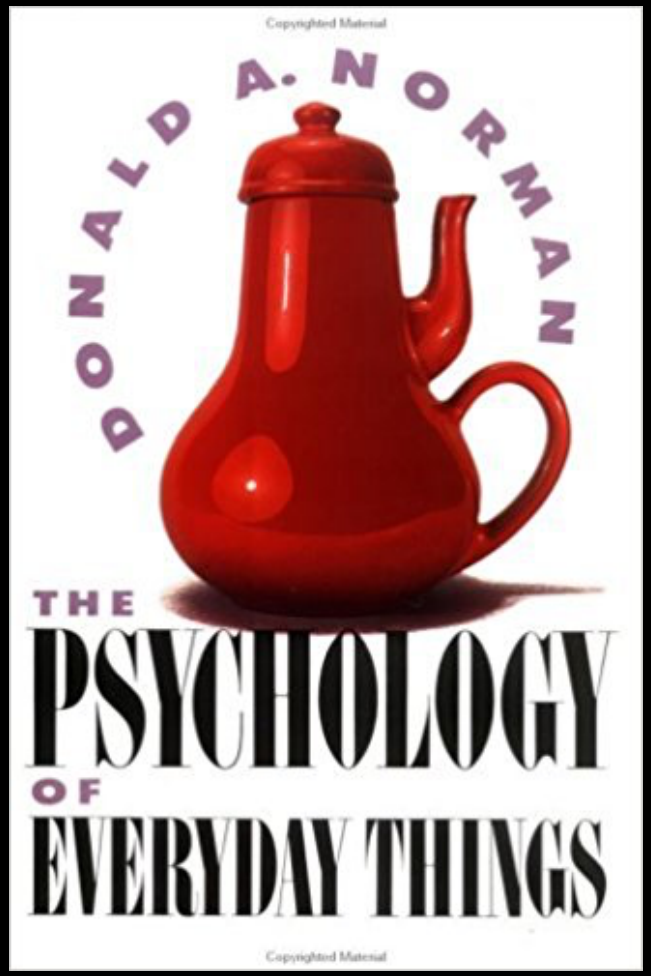
can design interfaces to match specific
user needs

can design interfaces, test them, and
interatively improve them

Today's topics

- **Heuristic evaluation:** method for systematically identifying and documenting usability problems

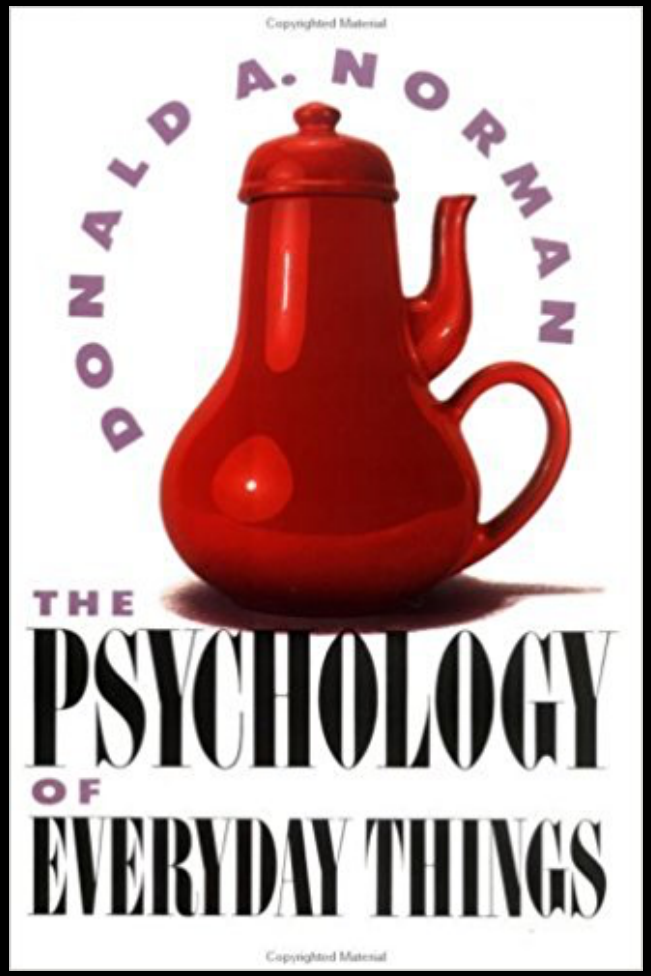
- **Norman:** provides useful *why* explanations for usability challenges



Today's topics

- **Heuristic evaluation:** method for systematically identifying and documenting usability problems

- **Norman:** provides useful *why* explanations for usability challenges



Heuristic evaluation

- Useful when you already know what the usability problems are
- Catch obvious bugs (so test participants can help with the more important issues)
- Systematic method that can help document *all* the usability bugs
- Encourages thinking through the underlying problem, then considering possible solutions

How to conduct a heuristic evaluation

1. Choose your heuristics (including custom heuristics)
2. Systematically analyze each component of the interface (e.g. screens) using your heuristics —screen by screen, heuristic by heuristic, task by task
3. Log each problem (can leave details for later)
4. Complete documentation for each problem —detailed description, screen shot, which heuristic is being violated?, severity (high, medium, low), suggested solution (maybe)
5. Prioritize issues and fix them (or pass them on to someone who will)

Nielsen's 10 heuristics

- Set of good practice heuristics that apply to many types of interactive systems
- But custom heuristics may work better for your design project

Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors

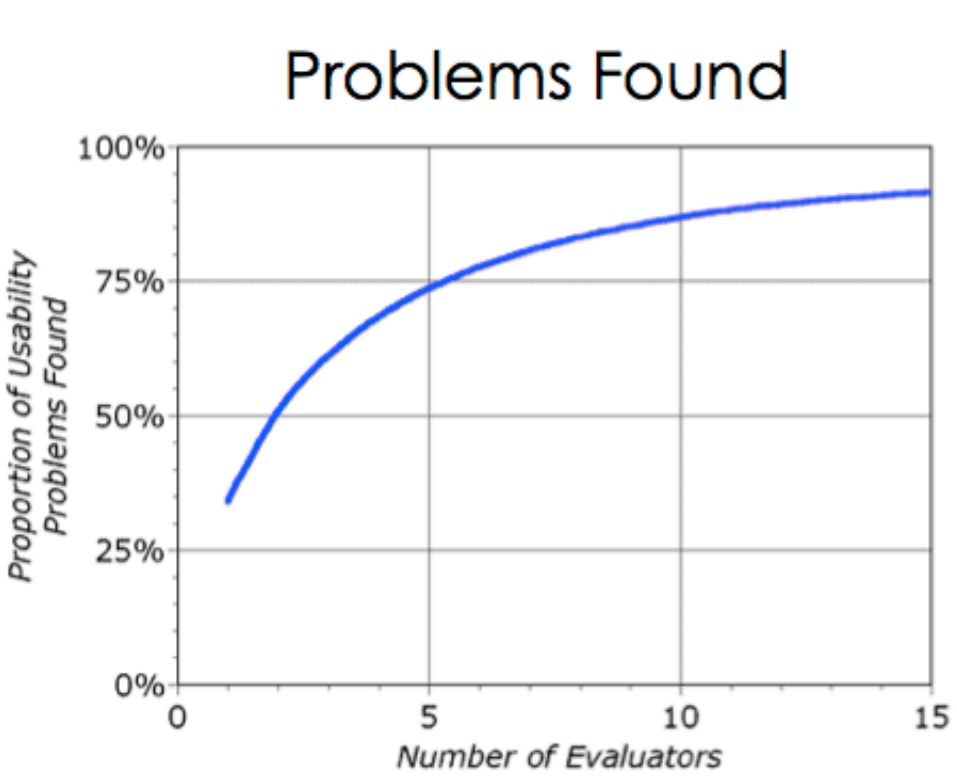
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

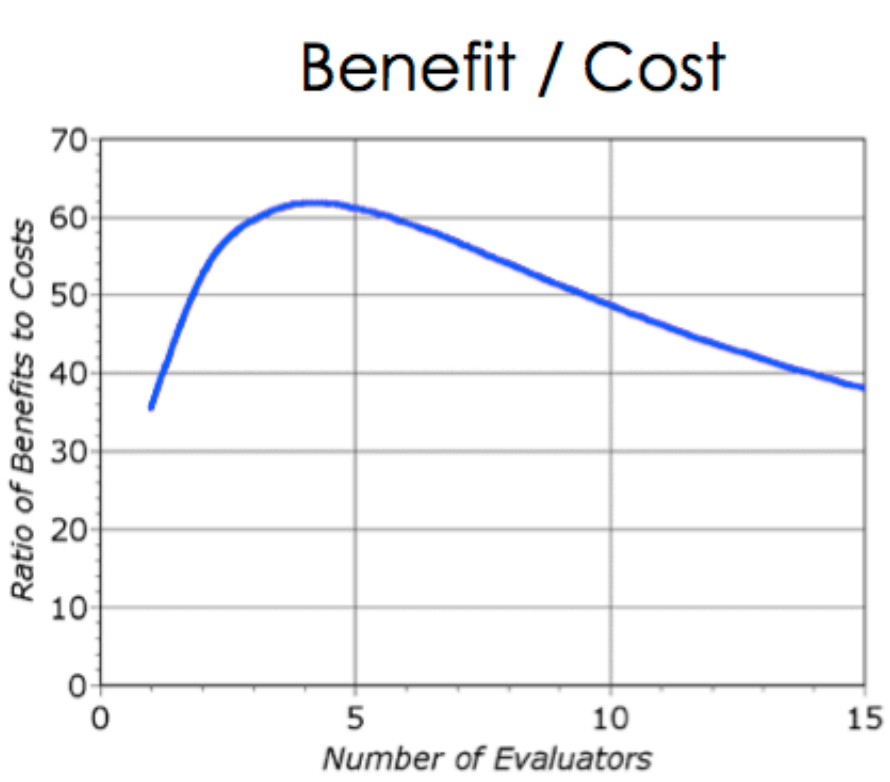
Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

How many evaluators?

- Nielsen recommends at least 3;
  5 even better





Worksheet

- Some people use worksheets to document heuristic evaluations (sometimes called Usability Aspect Report)
- Not necessary, but can help keep you on task
- See example on Moodle
- What's most important to track?
  —detailed description / reproduction steps, why it's a problem (what heuristic)

Usability Aspect Report Template
From Shaun Kane, based on UAR Template from Brad A. Myers and Bonnie John
http://www.cs.cmu.edu/~bam/uicourse/UARTemplate.doc

Complete this form *for each* problem or good aspect that you observe.

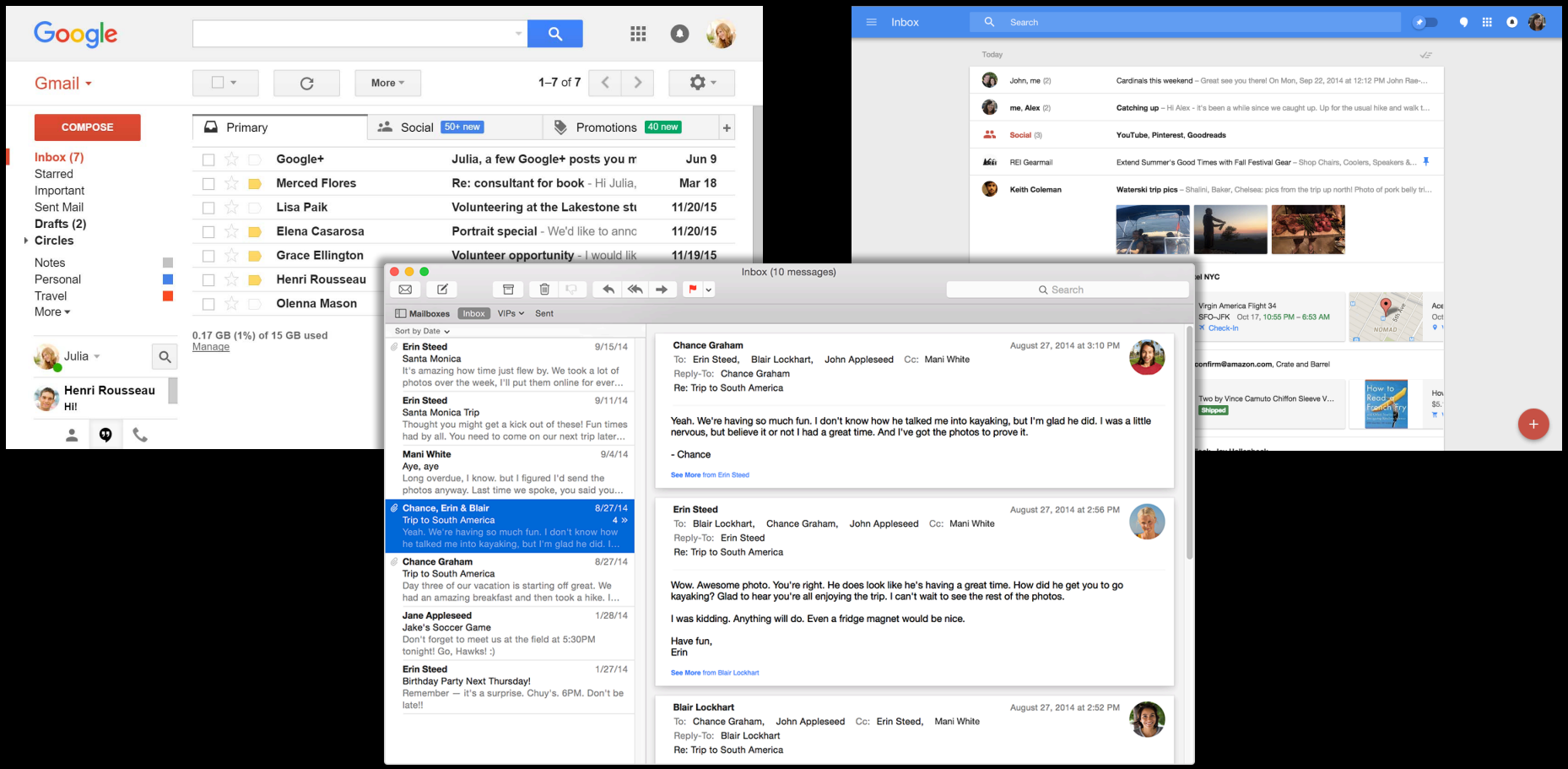
UAR #:	**Problem/Good:**	**Rated by:**
Name:		
Relevant heuristic:		
Steps to reproduce:		
Detailed explanation:		
Possible solution:		
Severity (low, medium, high, critical):	**See also:**	

What makes a good X?

- To get the most of this method, we may want to consider our own heuristics
- Can be based on study observations, end user comments, prior studies

Brainstorming heuristics

- Let's spend a few minutes brainstorming on usability heuristics for **email inboxes**
- How might we rate whether an email inbox is well designed?



Our email inbox heuristics

- Looks great
- Customized
- Sort and filter
- Overview
- Having folders
- Determining priority of messages
- Feedback
- Supporting multiple identities
- Easy access to features
- Quick actions
- Fewer clicks
- Skimmable (reading pane)
- Important things should be easy to find (“common tasks should be easy; rare tasks should be possible”)

Heuristics for your heuristics

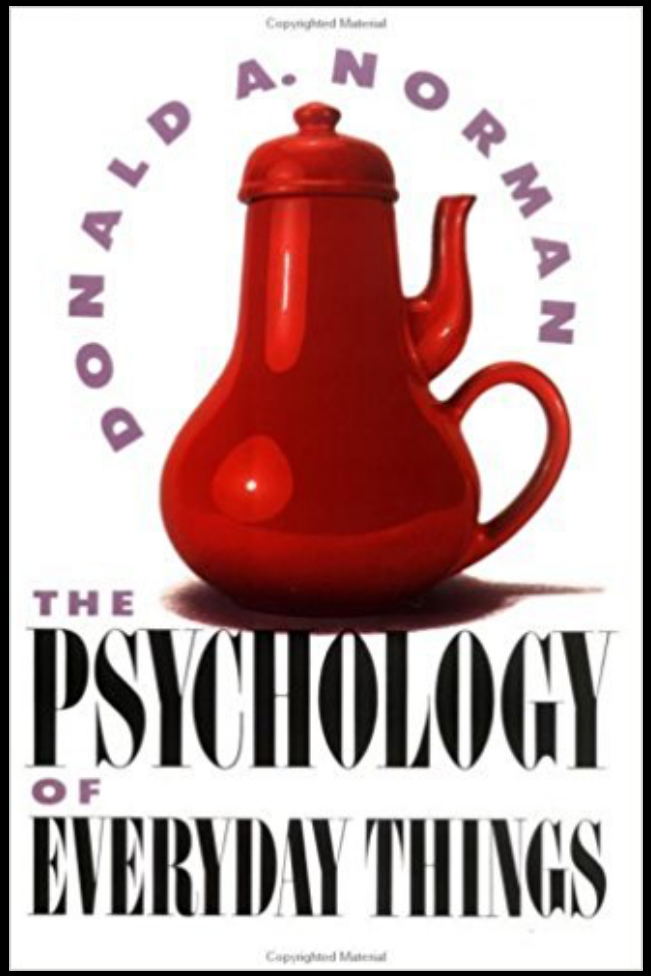
- (Separate out features)
- Plan to have other people use them
- Focus on the task
- Think about radical new designs

Questions about heuristic evaluation?

Today's topics

- **Heuristic evaluation:** method for systematically identifying and documenting usability problems

- **Norman:** provides useful *why* explanations for usability challenges



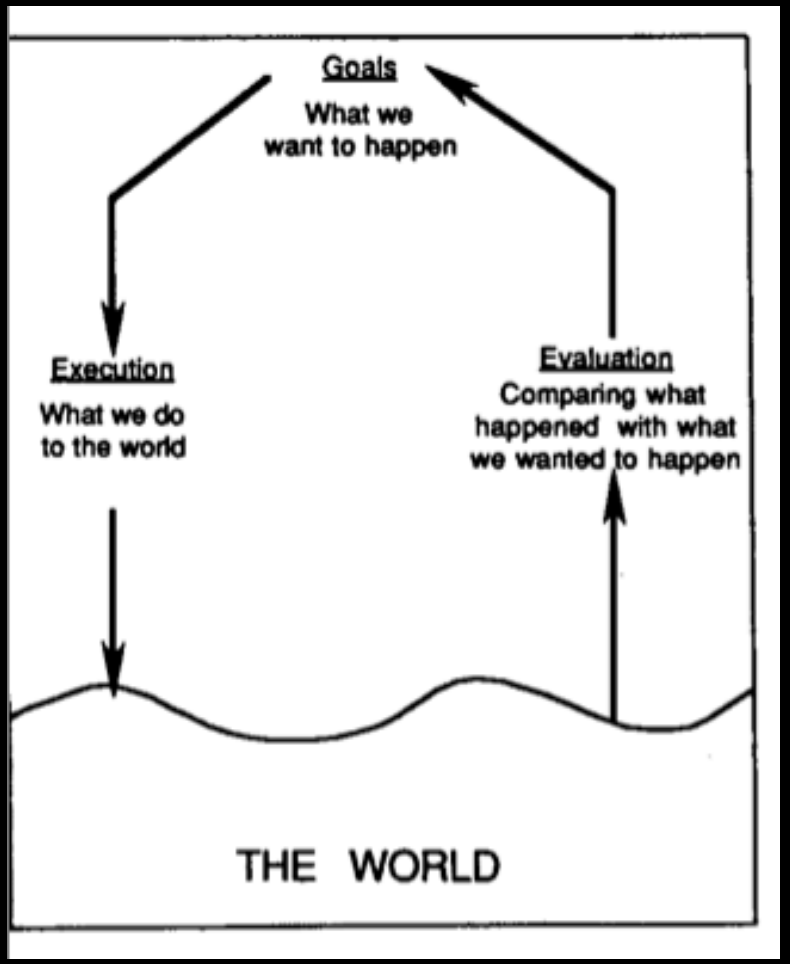
Norman: key ideas

- Gulfs of execution and evaluation
- Conceptual models
- Affordances
- Mapping
- Constraints
- Feedback

The Action Cycle

Gulfs of Execution and Evaluation

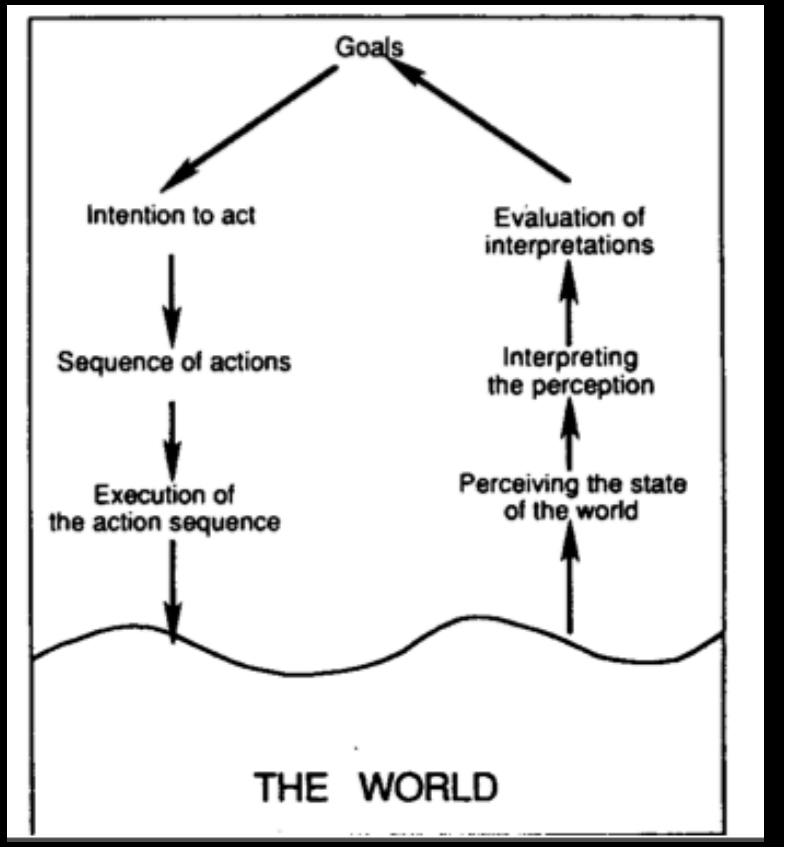
- Imagine you are a UI designer for an ATM manufacturer
- Your UX engineer comes to you with the feedback: *“our test participants couldn’t figure out how to withdraw cash”*
- … now what??



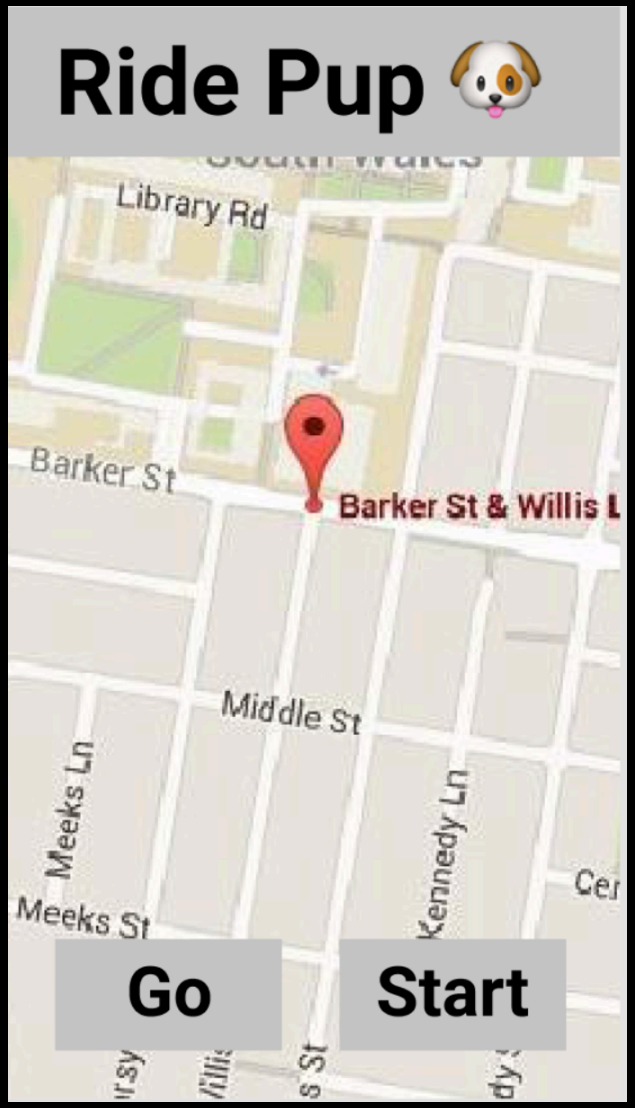
Norman's 7-stage model

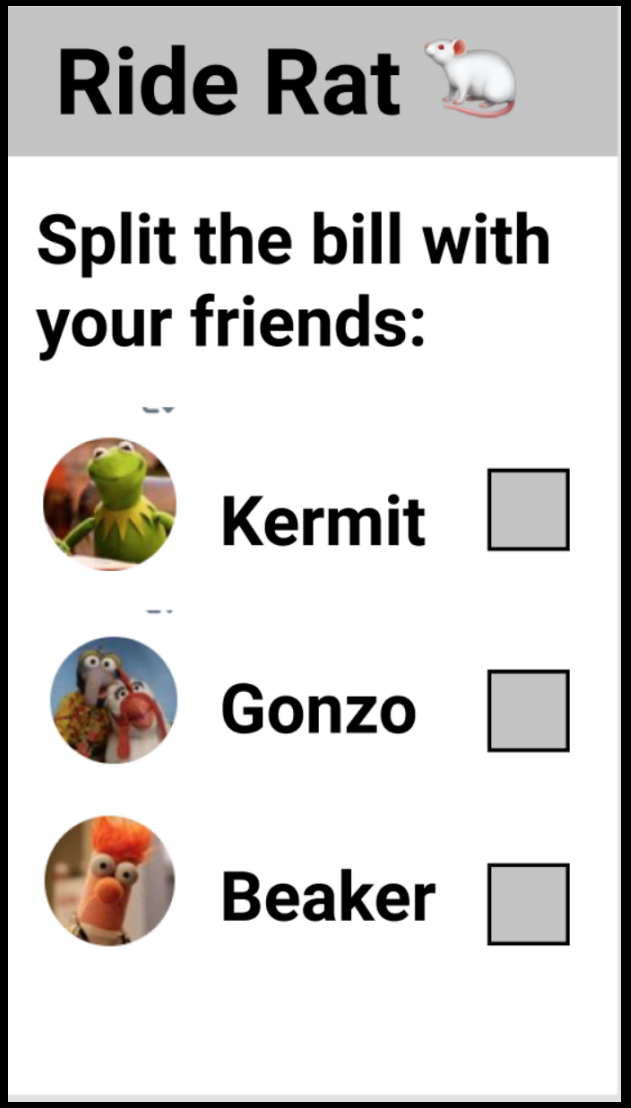
Gulfs of Execution and Evaluation

- Norman's 7-stage model gives us language to talk about when "users couldn't figure it out," act on it!
- Gulf of execution:
  —can't find it, can't figure out how to go to the next step
- Gulf of evaluation:
  — don't know whether it worked, don't know when to go to the next step



Execution and evaluation fails





Conceptual models

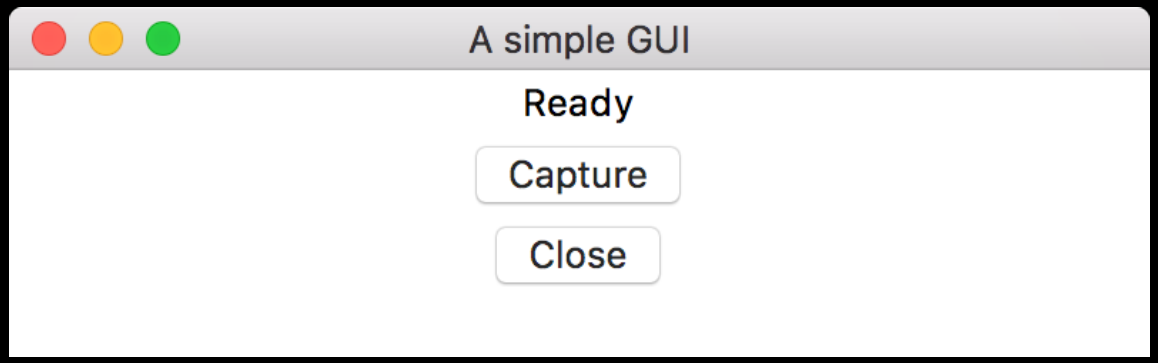
- What does the user think is happening?
- Human beings try to predict – what will happen if I do X?
  - We are “naïve scientists”
- Designers can influence the formation of the user’s conceptual model

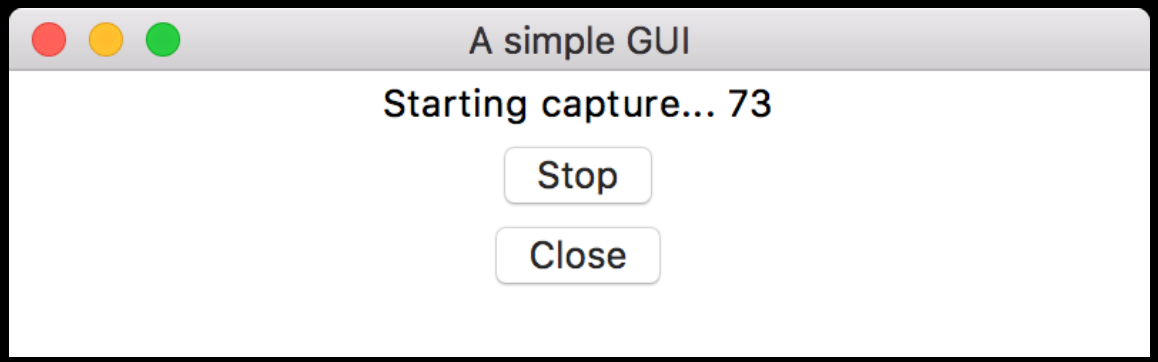
How models help

- The designer can present a simplified model (if it matches user behavior)

How models can hurt

- If user's conceptual model disagrees with how the system actually works, unexpected problems can occur






```
capture.onPress() {
while (1) {
// capture image in memory
images.append(capture)
print("Starting capture…" + i);
}
}

stop.onPress() {
foreach(img in images) {
img.save() // save to disk
}
}
```

How models can hurt

- If user's conceptual model disagrees with how the system actually works, unexpected problems can occur

Expectation:
when I press capture, it saves images until I close the program

Reality:
when I press capture, it saves images to a buffer. When I press stop, it saves them to disk

```
capture.onPress() {
while (1) {
// capture image in memory
images.append(capture)
print("Starting capture…" + i);
}
}

stop.onPress() {
foreach(img in images) {
img.save() // save to disk
}
}
```

Affordances, feedback, constraints, mapping

- Our tools to solve usability problems!

- Worth understanding their nuances
  —these terms have very specific definitions
  —important to be able to distinguish these

Affordances

- Structure action **without** instruction

History of affordances

- JJ Gibson, ants walking on a path
- What looks like very complex behavior is guided by the layout of the environment

These are NOT affordances

This is a cultural convention.

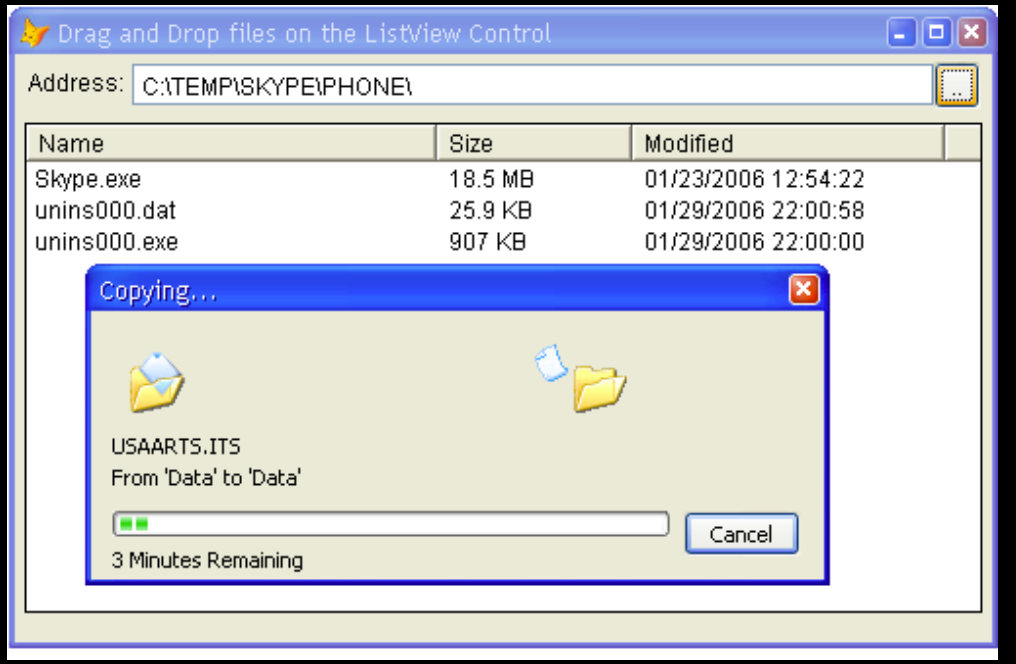
This is a perceived affordance

No! ---->

System status and feedback

- What just happened
- What is the system doing?
- Is it accepting input?
- What can I do from here?

System status and feedback



1 item added to Cart

Kindle Fire HD 7", HD Display, Wi-Fi, 8 GB...

$139.00

This is a gift

Why is this important?

Order subtotal: $139.00
1 item in your Cart

Edit your Cart | Proceed to checkout

Important messages about other items in your Cart

6 items in your Cart have changed price.

Items in your Shopping Cart will always reflect the most recent price displayed on their product detail pages.

- RunCore USB Enclosure For 1.8" SATA II LIF Solid State Drive (SSD). SATA II LIF (SSD) to USB Converter has decreased from $17.45 to $17.21
- 7 Wonders has increased from $33.17 to $33.43

Constraints

- We might frst think they’re bad, but they’re actually good

- Guide behavior and limit undesired actions

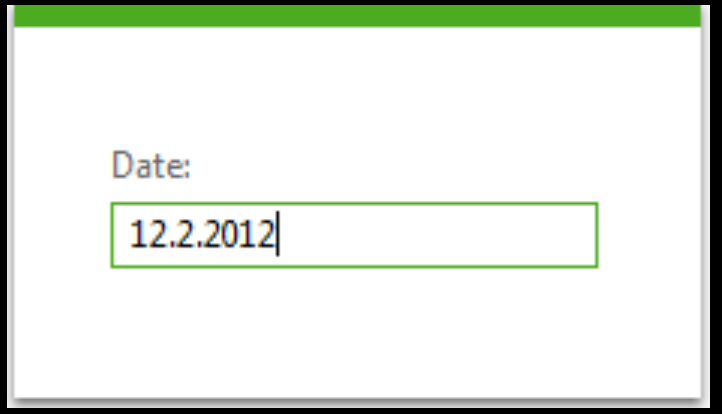
Constraints

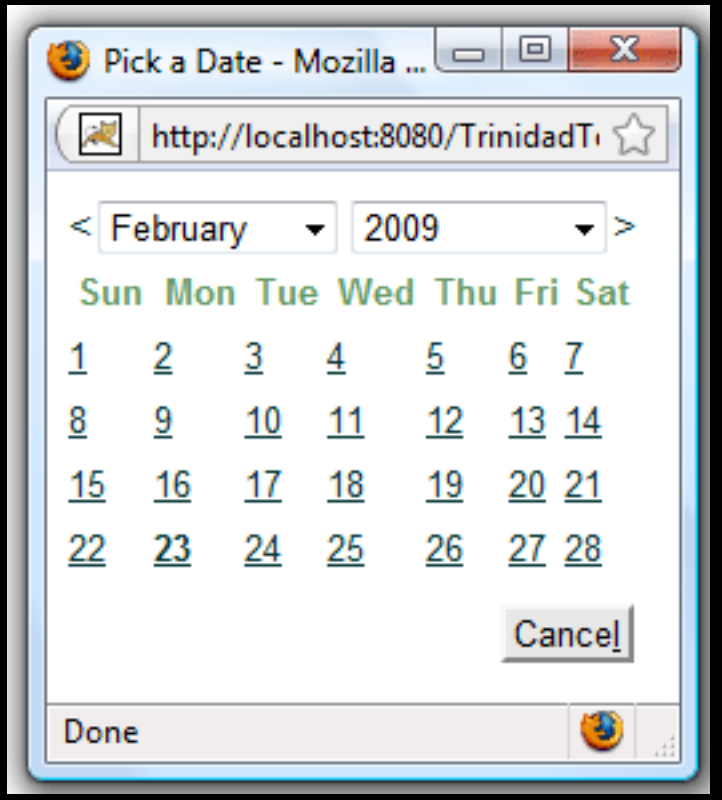
- Constraints limit **user actions**

- Can’t move between R and D without going through neutral

Constraints for date pickers

- What problems might occur with #1 but not #2?
- Can add error checking to #1 but then:
  - Date handling logic becomes much more complicated
  - Users can encounter frustrating errors
  - User knows they might make an error





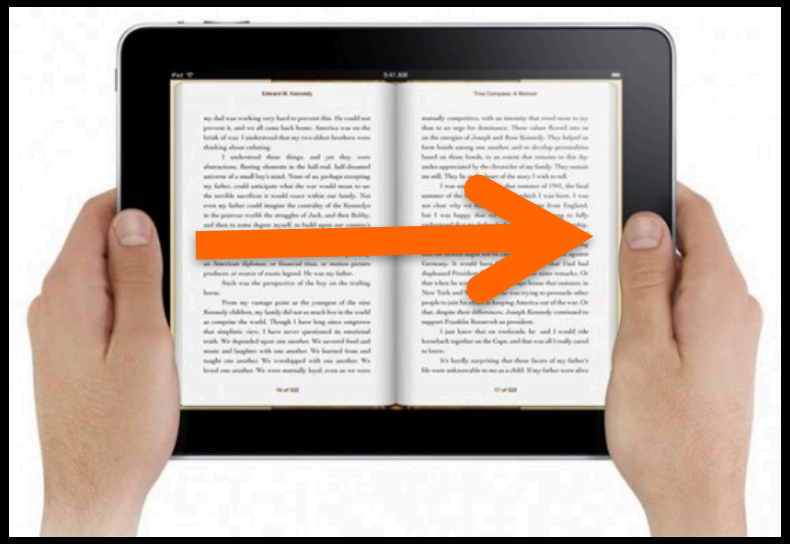
Mapping

- Relation between controls and the things that they control
- Good mappings logically map the input to the possible effects in a way that is logical
- Bad mappings are arbitrary or require instruction

1.13 Seat Adjustment Control from a Mercedes-Benz Automobile. This is an excellent example of natural mapping. The control is in the shape of the seat itself: the mapping is straightforward. To move the front edge of the seat higher, lift up on the front part of the button. To make the seat back recline, move the button back. Mercedes-Benz automobiles are obviously not everyday things for most people, but the principle doesn't require great expense or wealth. The same principle could be applied to much more common objects.

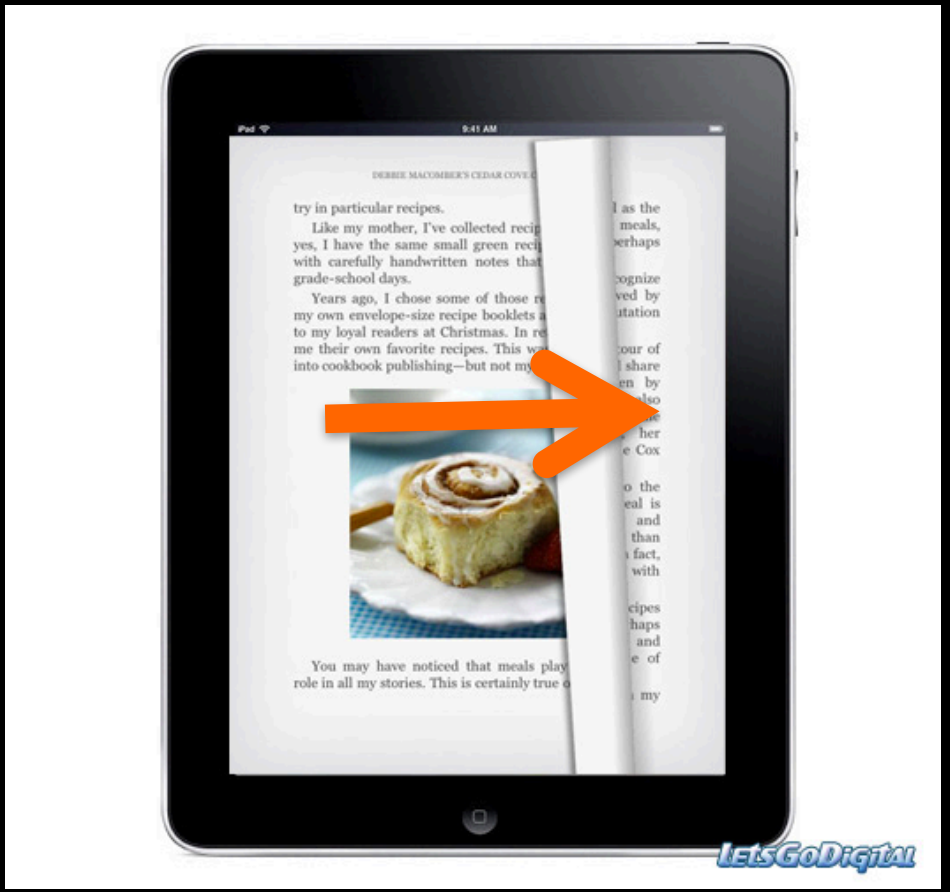
- Users sometimes tacitly know mappings without realizing it!

Mapping for gestures



What happens when I gesture in this direction?

Mapping for gestures



What happens when I gesture in this direction?

But what does Heidegger think?

- Philosopher Martin Heidegger in *Being and Time* (1927)*:*
- *"The peculiarity of what is proximally [present] to hand is that, in its readiness-to-hand, it must, as it were, withdraw … that with which we concern ourselves primarily is the work …"*
- **Ready-to-hand** – expert use, focus on the task and don't even think about the tool
- **Present-at-hand** – unskilled use, thinking directly about manipulating the tool

Activity for today

- Find examples of the following in the classroom: **Affordances, conventional signifiers, system status/feedback, constraints, mapping**
- Make a list of the item and an explanation of why it is that item
- Try to come up with as many (confident) examples as you can!