Review

- What are the Seven Stages of Action?
  - Where are gulfs between the seven stages?
  - What are design implications?

- Why is it important to recognize the differences between knowledge in the world and in the head?

- Why is the waterfall model not suitable for developing interactive systems?
  - What is the alternative?
  - Why is the DIA cycle more superior than the Waterfall Model?

- What are the first two questions to ask when designing an interactive system?
  - What tools do we have to answer those questions?

Errors

- People make errors using everyday objects all the time
- Often blame themselves (untypical!)
- Often caused by taught helplessness
  - E.g., maths classes
- May lead to learned helplessness
  - Conspiracy of silence, depression
- Not only “dumb folk” have misconceptions of everyday life, and often those “wrong” models work better for everyday life
  - E.g., thermostats

Mistakes

- Form wrong goal, then execute action sequence
- Hard to detect
- Often major events
- Result of conscious decision/thinking
- Reasons: leaping to wrong conclusions, false causalities

Slips

- Most everyday errors
- Small things going wrong
- Goal formed, but execution messed up
- Usually easy to discover
- Occur mostly in skilled behavior
- Often caused by lack of attention, busy, tired, stressed, bored, more important things to do,…
- We can only do one conscious thing at once
  - Jef Raskin, The Humane Interface: Walking and eating and solving a maths problem
Types of Slips

- **Capture errors**
  - Two action sequences with similar initial but different later sequence
  - The one well practiced can easily “capture” the unfamiliar one
  - E.g., driving somewhere on Sunday, then taking the wrong turn to “go to work as usual”. Or, pocketing a borrowed pen.

- **Description errors**
  - Intention not described in enough detail, allowing 2 different action sequences to fit it
  - Often occur if similar objects are physically close to each other (e.g., switches)
  - E.g., throwing t-shirt into toilet instead of laundry basket
  - Putting a lid onto the obviously wrong container
  - Pouring orange juice into your coffee pot

- **Data-driven errors**
  - Arriving sensory data intrudes into ongoing action sequence, causing unintended behavior
  - E.g., dialing a room number instead of phone number because you look at the room number on a sign in front of you when dialing

- **Associative activation errors**
  - Internal association triggers wrong action
  - Also known as Freudian slips
  - E.g., answering the phone saying “Come in!”

- **Loss-of-activation errors**
  - Forgetting goal while action sequence is running
  - Special version of forgetting to do something
  - E.g., walking into your bedroom then wondering what you wanted to do here.
  - Can be reactivated by repeating original stimulus
  - E.g., walking back to the living room where you see something that reminds you why you were going to your bedroom

- **Premature conclusion errors**
  - Forgetting to complete action sequence because main part of goal is accomplished
  - E.g., ATM card in machine, originals in copier
Types of Slips

- **Mode errors**
  - Triggering the wrong action because the device is in a different mode than expected
  - Who has seen this in their favorite text editor: ":wq"?
  - Happens whenever devices resort to modes to cope with more functions than controls
  - The most prominent problem in many software user interfaces

In-Class Exercise: Slips

- In groups of two, think of three examples of slips that happened to you. What type are they?
  - Capture (driving to work)
  - Description (shirt in toilet)
  - Data-driven (dial room number)
  - Associative Activation ("Come in!")
  - Loss of Activation (walk into bedroom)
  - Premature Conclusion (copier)
  - Mode (vi)

Detecting Slips

- Easy but requires visible feedback
  - Example: "Adjust the window!"
- Problem: Finding the right level at which to correct
  - Are we doing this bottom-up?
  - The wrong car key
  - Confirmation is unlikely to catch errors
    - "Remove file bla.txt!"
  - Soft, reversible actions are better (e.g., trashcan), but people begin to rely on it

Decision Structures

- To reduce chance of error, use either shallow or narrow decision trees
  - **Shallow**: No planning required, e.g., ice cream parlor menu
  - **Narrow**: No deep thinking required, e.g., cook book instructions, start your car, motorway exits
- **Wide and deep structures**:
  - Games like chess, etc.
  - Designed to occupy the mind
  - Subconscious thought is effortless, associative, pattern-matching
  - Conscious thought is slow, serial, demanding
Designing for Error

- Assume all possible errors will be made
- Minimize the chance of errors occurring
- Minimize their effect if they are made
- Make them easy to detect
- Make them easy to reverse (undo)
- Watch people using your system (and their slips and mistakes)
- Don’t punish, don’t ignore
- Warning signals are ignored, warning features bypassed if inconvenient

Forcing Functions

- Forcing functions can help to avoid errors (= Extreme physical constraints)
- Think through the burden on normal operation!
  - E.g., seat belts
- Known from safety engineering
  - **Lockout** prevents an action
    - E.g., stairways to basements
  - **Lockin** prevents prematurely stopping an action
    - E.g., soft power-off switch on computers to avoid data loss
  - **Interlock** enforces correct sequence
    - E.g., microwave turning off when opened, shelf in public restrooms
Why Is Good Design So Rare?

- Pressure of product schedules, creeping featurism
- Curse of individuality, “being different”
  - Swedish Hair Dryer
- Good design takes many iterations, but after initial failures many products are “dead”

How Not to Design a GUI

- Make things invisible, widen the gulfs
- No feedback
- Use non-obvious commands and arbitrary mappings between them and outcomes
- Use tech speak and abbrv.
- Be impolite, especially in error messages
- Make operations potentially fatal. No Undo.

Designing for Users

- Form should follow function
- Designers are virtually never users
- Clients are not always users
- There is no average user
  - Designing it “right for 99% of people in the US” leaves out 2.5 million Americans
  - Age effects set in in the mid-20s

Seven Principles of Design

- Use knowledge in the world and in the head
- Simplify task structures
- Make things visible, bridge the gulfs of execution and evaluation
- Use natural mappings
- Use natural and artificial constraints
- Design for error
- When all else fails, standardize
Closing the Book

- Norman’s book is outdated in many technology aspects
- But it has hopefully provided you with a new power of observing people and their interaction with everyday objects and technology
- It will be a book to go back to and re-read in a few years
- Read through the rest of the book this week!

Prototyping

Theory
- ✓ Models of interaction
- ✓ Affordances, mappings, constraints, types of knowledge, errors
- Design principles
- Human cognition and performance
- History and vision of HCI

Practice
- ✓ Sketching
- ✓ User observation
- ✓ Iterative design
  ➞ Prototyping
  • Ideation
  • User study and evaluation
When to Prototype

Prototype: proof of concept

Design

Analyze

Implement

Prototyping in DIA iterations

Project start

D: Brainstorm different representations
D: Choose a representation
D: Rough out interface style

A/D: Task centered walkthrough&redesign
A/D: Fine tune interface, screen design
A/D: Heuristic evaluation and redesign
A/D: Usability testing and redesign

I: Low fidelity paper prototypes
I: Medium fidelity prototypes
I: High fidelity prototypes
I: Working systems

Project End

Paper Prototypes

• First prototypes, quick and cheap

• You can use storyboards as your first prototype!

• Rough paper & pencil sketches of interface or central UI dialogs

• Hand-drawn, no ruler, no computer!

• Type A: Storyboard-like
  • Put several frames with sketched snapshots of the UI on one page
  • Label each frame and each connection
  • Only allows you to show one fixed interaction sequence (scenario)
  • Like a storyboard, but only shows the UI (and maybe the user’s hand), not the entire environment of the task

Paper Prototype Example: Shopping Application

• Uses a storyboard-like format

• Includes two sample interaction sequences (scenarios)

• Bad example because it is not hand-drawn
Paper Prototypes

- Type B: Flipbook
  - Sketch each UI snapshot frame on separate page
  - Collect in a loosely bound flipbook that flips over easily
  - Usage: Show start screen page to user—he selects an action—turn to the resulting page from your flipbook, etc.
  - Allows you to simulate the UI for a user (Wizard Of Oz)

- Pro: Not detailed, so designer and user focus on important high-level UI design

- Con: Dialog sequence hard to convey unless you drive it yourself (as in the flipbook); drawing many screens is a lot of work
Post-It Prototype

- More interactive paper prototype
- Dialogs, menus, windows on post-it notes in multiple layers
- Allows simulating opening dialogs, etc., by manipulating notes
- Quick to change by making new notes
- Tip: Create empty templates for dialog objects, then fill in
- Tip: Videotape user session for later analysis
Expandable lists
Images: Paper Prototyping by Carolyn Snyder, 2003

Disabled ("grayed-out") controls
Images: Paper Prototyping by Carolyn Snyder, 2003

Simulating touchscreen widget with paper prototype
Kaiser, Dietert. DIS1 students in 2010

Software Prototype

- Medium fidelity prototype
- More detailed, more precise, interactive
- Create only after initial, simpler (paper) prototypes!
- Mock-up (model, illusion) of some (but not all) aspects of the final UI
- Example: Flash animation
- Important: UI, not functionality, is key!
- Pro: More engaging for user to try, user can play with it without designer around
Software Prototype: Dangers

- Users focus on design details and overlook larger problems
- Users afraid to criticize or suggest changes to “nice” UI design
  - Looks like it was so much work…
- Management may think it’s real 😊
  - Looks like the software is almost done
  - Reason: Conceptual models

How to Limit Prototypes

- Vertical prototype
  - Few functions, but those implemented in detail
  - Allows testing general design ideas by example
- Horizontal prototype
  - Entire UI visible, but no functionality
  - Simulate each interaction step (nothing “works”)
- Scenario
  - Combination of horizontal and vertical prototype
  - Script simulates only fixed interaction paths

Software Protootyping: Screenshots

- Photoshop, PowerPoint, etc.
- Draw screens / UI storyboards
- Thin horizontal prototype
- Easier to change than hand drawings
- Allows for visual detail and quality
- Designs can become part of actual UI
  - Useful for non-standard GUIs
- Easy to distribute electronically

Screenshots: Problems

- No interaction, does not capture any dynamic behavior or “feel” of the UI
- Danger of looking too polished, limits feedback, suggests the interface is “done”
- Missing physical aspects of devices
Software Prototyping: On-Screen Storyboards

- Scripted simulations
- Using media tools such as PowerPoint or Photoshop layers
- More potential for interactivity:
  - Scene transition by simple input, timing, animation
- Prototype with slightly more vertical depth
- Use as click-through prototype or for pitching
- Pro: looks real, good for non-standard UIs, no programming
- Con: simulation fails when script is not followed
Wizard of Oz

- Human ‘wizard’ simulates system response
  - Interprets user input according to an algorithm
  - Controls computer to simulate appropriate output
  - Uses real or mock interface
  - Wizard sometimes visible, sometimes hidden
    “Pay no attention to the man behind the curtain!”

- Good for:
  - Adding simulated and complex vertical functionality
  - Testing futuristic ideas
  - Example: 1984 IBM voice recognition editor

Image: Buxton 2007, Sketching User Interfaces
Hardware Prototype

- For systems that are hard to imagine by software alone
  - Example: Palm’s wooden blocks

- Physical interaction is important
  - E.g., new 3-D mouse

- Design in wood, foam core, plastics, styrofoam, cardboard, …

- Problem: high effort to build and change

What to Do with a Prototype

- Throw away
  - If creation was quick and cheap

- Continue to develop
  - Prototype improved incrementally
  - Becomes final product
  - Problem: Has to use production-strength technology (i.e., generally not Flash…)

Summary

- “To err is human”
- Slips are small errors that mess up task executions
  - Good design can prevent slips or help recover from them
- Creeping featurism causes bad designs to recur
- Prototypes let you catch big bugs early
- Choose prototyping method to match what you wanted to test