Usability Design Principles

CSE 403

Announcements

- Team mailing lists
  - cse403-t1@cs - Busta' Sandwich
  - cse403-t2@cs - Online Jam Space
- View your LCO grade
  http://www.cs.washington.edu/education/courses/cse403/05wi/grading403.html

Outline

- Why usability is important
- Design of Everyday Things
- User-centered design
- Task analysis
- Contextual inquiry
- Prototyping
- Group meetings (time permitting)

Resources

- Lecture from spring 2005 (Richard Anderson)
- Lectures from fall 2004 CSE 490JL (James Landay)

Why bother with UI design

- Think of an application whose UI frustrates you
  - What frustrates you about it?
  - Why do you think it was designed that way?
  - Would it have helped the designers to have asked you?

Consequences of bad UI design

- Frustrations
  - Errors
  - Hard to find functionality
  - Time-consuming
  - Distracting
- Why was it designed that way
  - Didn't care
  - Didn't have enough time
  - Didn't know better
  - Didn't get enough of the right kind of feedback
- Would asking you (an end-user) help?
  - Of course, but with caveats

Tradeoffs

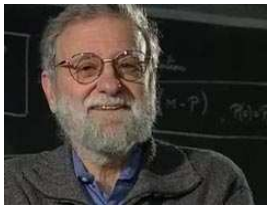
- Recognize that there are engineering tradeoffs
- Design is hard
  - It usually takes about five or six attempts to get a product right
  - Vast number of variables

Design

- How do people interact with computers?
  - Tremendous flexibility in designing/building interactions
- Look at physical objects
  - Thousands of years of design experience
  - Human side is the same

Design of Everyday Things

- Don Norman
  - Cognitive Scientist
  - Apple Fellow
  - Prolific writer
- Basic theme
  - Understand how common objects are used



Design examples

- Doors
  - Basic requirement – a user must be able to open the door and walk through it
  - What could go wrong?
  - Lack of visual cues

Saigon Deli – U. District

Telephones

- Basic dial / number pad is standard
- Mechanisms for additional functionality can be difficult
  - Arbitrary
  - Multifunction keys
  - No mental model

Stove Top

Conceptual models

- Mental model of how things work
  - Does not need to be correct, just predictive
- Don Norman – refrigerator / freezer temperature control
- Thermostats



Affordance

- Perceived and actual properties of an object – especially the properties that determines how an object is used
  - A door *affords* going through
  - A chair *affords* sitting on
  - Glass *affords* seeing through (or breaking)
- Doors – indication of how to open them
- Light switches – indication of function

The principle of mapping

- Mental association between objects and actions
- Some natural
- Some cultural
- Some arbitrary

The principal of feedback

- Indication that an operation is taking place
- Key clicks
- Direct physical response when opening a door
- Hour glass cursor on a long operation

Cognitive Load

- How little memory do we need?
  - Short term memory
  - Long term memory
- Avoid requiring arbitrary information
  - Visual information
    - Labels, Groupings, Mappings
  - Conventions
  - Transfer
    - Common experience
    - Conceptual models

Designing for Failure

- Design for fallible users
- Understand classes of errors
- Error minimization
- Error prevention
- Error mitigation
- Error recovery

Errors

- What is an error?
- What kinds of errors can be accommodated for by better design?
- Example: Car related
  - I drive with my high beams on
  - I misuse the controls in an unfamiliar car in a pressure situation
  - I lock my keys in the car
  - I take the wrong exit off the freeway

Beginners, Experts, Intermediates

- Who are you designing for?
- How do your users work?
  - task analysis, interviews, and observation
- How do your users think?
  - understand human cognition
  - observe users performing tasks
- How do your users interact with UIs?
  - observe!

Task Analysis

- Who is going to use the system?
- What tasks do they now perform?
- What tasks are desired?
- How are the tasks learned?
- Where are the tasks performed?
- What's the relationship between user & data?
- What other tools does the user have?
- How do users communicate with each other?
- How often are the tasks performed?
- What are the time constraints on the tasks?
- What happens when things go wrong?

Who?

- Identity
  - in-house or specific customer is easy
  - need several typical users for broad product
- Background
- Skills
- Habits and preferences
- Physical characteristics
  - height? vision?
  - hand-eye coordination
  - stamina

Involve Users to Answer Task Analysis Questions

- Users help designers learn
  - what is involved in their jobs
  - what tools they use
  - i.e., what they do
- Developers reveal technical capabilities
  - builds rapport & an idea of what is possible
  - user's can comment on whether ideas make sense
- How do we do this?
  - observe & interview prospective users in their context!

Contextual Inquiry

- Way of understanding users' needs and practices
- Master / Apprentice model allows customer to teach us what they do!
  - master does the "work" & talks about it while "working"
  - we interrupt to ask questions as they go
- The Where, How, and What expose the Why

Principles of CI

- Context
  - go to the workplace & see the work as it unfolds
  - people summarize, but we want details
    - keep it concrete when people start to abstract
      - "We usually get reports by email", ask "Can I see one?"
- Interpretation
  - facts are only the starting point, design based on interpretation
  - validate & rephrase
    - share interpretations to check your reasoning
      - Ex. "So accountability means a paper trail?"
    - people will be uncomfortable until the phrasing is right
      - be committed to listening ("Huh?", "Umm…", "Yes, but…")

Principles (cont'd)

- Focus
  - interviewer needs data about specific kind of practices
    - "steer" conversation to stay on useful topics
  - respect triggers (flags to change focus)
    - shift of attention (someone walks in)
    - surprises (you know it is "wrong")

Interviewing

- Use recording technologies
  - notebooks, tape recorders, still & video cameras
- Structure
  - conventional interview (15 minutes)
    - introduce focus & deal with ethical issues
    - get used to each other by getting summary data
  - transition (30 seconds)
    - state new rules – they work while you watch & interrupt
  - contextual interview (1-2 hours)
    - take notes, draw, be nosy! ("who was on the phone?")
  - wrap-up (15 minutes)
    - summarize your notes & confirm what is important
- Master / apprentice can be hard
  - e.g., sometimes need to put down your company

What they might say

- "This system is too difficult"
- "You don't have the steps in the order we do them"
- Do not take comments personally
  - you shouldn't have a personal stake
- Be careful not to judge participants
- Goal is to make the system easy to use for your intended users

Using what you learn from Contextual Inquiry

- Rough out an interface design
  - discard features that don't support your tasks
    - or add a real task that exercises that feature
  - major screens & functions (not too detailed)
  - hand sketched
- Produce scenarios for each task
  - what user has to do & what they would see
  - step-by-step performance of task
  - illustrate using storyboards
    - sequences of sketches showing screens & transitions

Prototyping

- Experiment with alternative designs
- Get feedback on our design faster
  - fix problems before code is written
  - saves money
- Sketching is easy, fast, and good for getting feedback
- Keep the design centered on the user
  - must test & observe ideas with users

Pitfalls to avoid when doing user-centered design

- Users are not always right
  - cannot anticipate new technology accurately
  - job is to build system users will want
    - not system users *say* they want
    - be very careful about this (you are outsider)
      - if you can't get users interested in your hot idea, you're probably missing something
- Design/observe forever without prototyping
  - rapid prototyping, evaluation, & iteration is key

Better design

- Task analysis
- Contextual Inquiry
- Prototype early, often
- Principles to remember
  - Provide affordances, feedback
  - Match users' mental models
  - Keep cognitive load small.
- Keep users in the loop!