User Interface Design, Prototyping, and Evaluation

Introduction and Overview

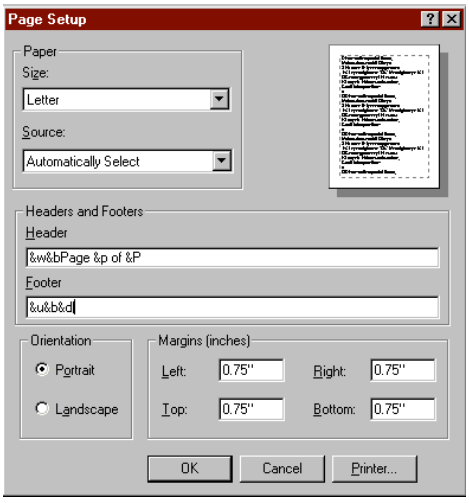
Jason I. Hong

July 06, 2005

Symposium on Usable Privacy and Security

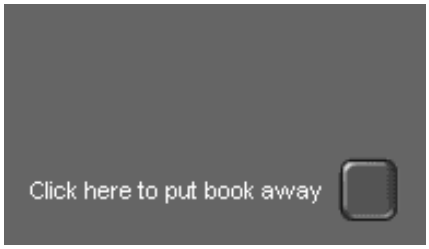
Carnegie Mellon Human-Computer Interaction Institute

User Interface Hall of Fame or Shame?

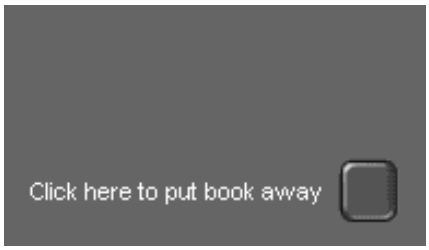


- IE5 page setup for printing

User Interface Hall of Fame or Shame?



- From IBM's RealCD
  - prompt
  - button



- From IBM's RealCD
  - prompt
  - button

Doors with “Pull” Signs

User Interface Design, Prototyping, and Evaluation

Introduction & Course Overview

Jason I. Hong
July 6, 2005

Outline

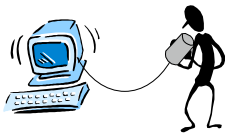
- Who am I?
- Human-Computer Interaction Introduction
- Course overview & schedule
- Introductions

Who am I?

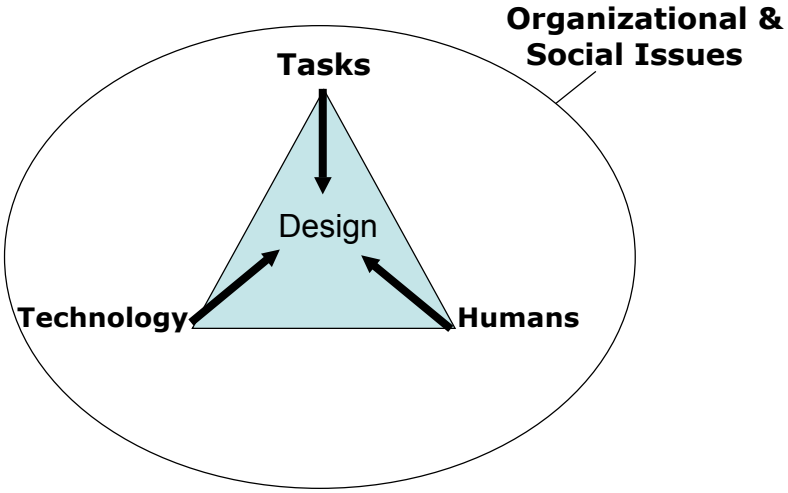
- Jason Hong
  - Assistant Professor in HCI at Carnegie Mellon
  - BA in CS & Math from Georgia Tech
  - Ph.D. from University of California, Berkeley
  - Human-Computer Interaction (HCI)
    - Privacy, location-aware computing, mobile computing

Human-Computer Interaction (HCI)

- Human
  - the end-user of a program
  - the others in the organization
- Computer
  - the machine the program runs on
  - clients & servers, PDAs, microwaves
- Interaction
  - the user tells the computer what they want
  - the computer communicates results



HCI Approach to UI Design



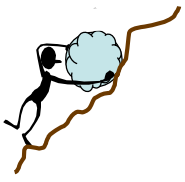
- Other considerations we won't look at
  - Business models, level of fun

Why is HCI Important?

- Major part of work for “real” programs (~50%)
- Bad user interfaces cost:
  - money
    - 5%↑ satisfaction –> up to 85%↑profits
    - finding problems early makes them easier to fix
  - reputation of organization (e.g., brand loyalty)
  - time (wasted effort, wasted energy)
  - lives (Therac-25)

Why is HCI Important?

- Privacy and Security
  - phishing scams
  - accidental disclosures (ex. location info, cookies)
  - difficulty diagnosing the situation (intrusion detection)
  - intentionally circumventing security mechanisms
- User interfaces hard to get right
  - people are unpredictable
  - intuition of designers often wrong



Myths about Good Design

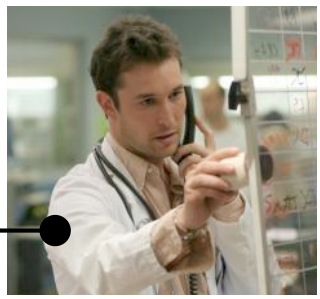
- Myth 1: Good design is just common sense
  - why are there so many bad web sites? hard to use apps?
- Myth 2: Only experts create good designs
  - experts faster, this course is on simple and effective techniques anyone can apply
- Myth 3: We can fix the user interface at the end
  - good design is more than just user interface
  - having right features, building those features right
- Myth 4: Good design takes too long / costs too much
  - simple and effective techniques that can reduce total development time & cost (finds problems early on)
- Myth 5: Good design is just cool graphics
  - graphics part of bigger picture of **what** to communicate & how

Who Builds User Interfaces?

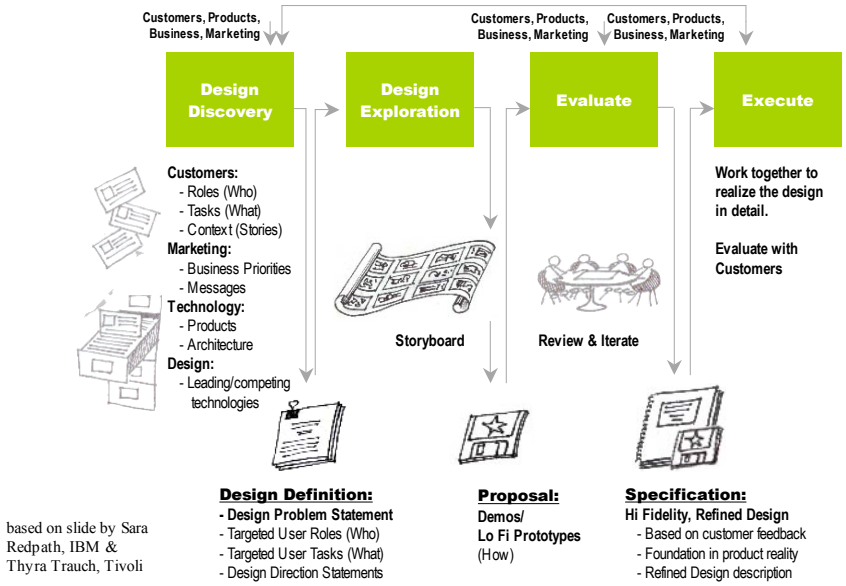
- A team of specialists (ideally)
  - graphic designers
  - interaction / interface designers
  - information architects
  - technical writers
  - marketers
  - test engineers
  - usability engineers
  - software engineers
  - users

How to Design and Build UIs

- User interface design process
- Usability goals
- User-centered design
- Task analysis & contextual inquiry
- Rapid prototyping
- Evaluation
- Programming



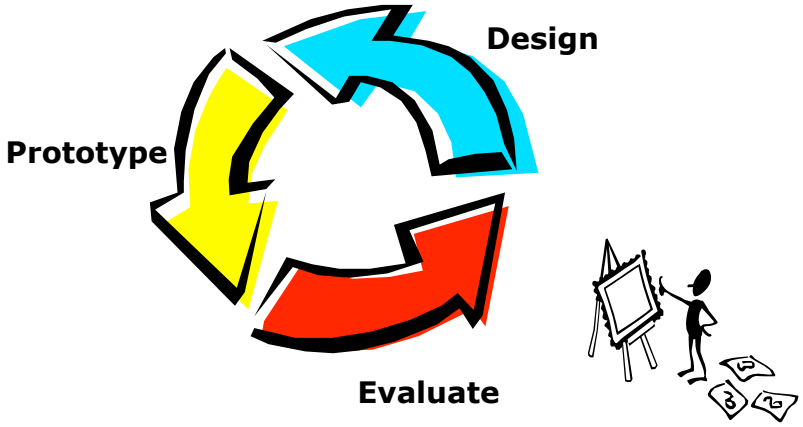
User Interface Development Process



based on slide by Sara Redpath, IBM & Thyra Trauch, Tivoli

Iteration

At every stage!



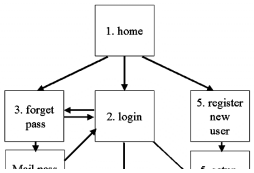
Design

- Design is driven by requirements
  - what the artifact is for
  - not how it is to be implemented
  - e.g., PDA not as important as “mobile” app.
- A design represents the artifact
  - for UIs these representations include (?)
    - screen sketches or storyboards
    - flow diagrams/outline showing task structure
    - executable prototypes
  - representations simplify

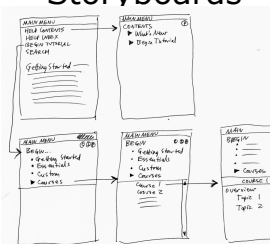
Write essay
start word processor
write outline
fill out outline
Start word processor
find word processor icon
double click on icon
Write outline
write down high-level ideas
.
.
.

Web Design Representations

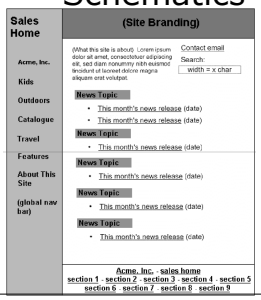
Site Maps



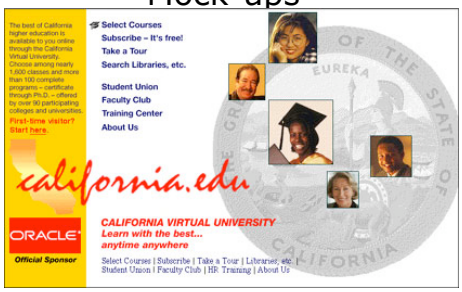
Storyboards



Schematics



Mock-ups



Usability Goals?

According to the ISO:

The *effectiveness*, *efficiency*, and *satisfaction* with which specified users achieve specified goals in particular environments

International Organization for Standardization

- This does not mean you have to create a "dry" design or something that is only good for novices – it all depends on your goals

Usability Goals

- Set goals early & later use to measure progress
- Goals often have tradeoffs, so prioritize
- Example goals:

- Learnable
  - faster the 2nd time & so on
- Memorable
  - from session to session
- Flexible
  - multiple ways to accomplish tasks

- Efficient
  - perform tasks quickly
- Robust
  - minimal error rates
  - good feedback so user can recover
- Pleasing
  - high user satisfaction
- Fun

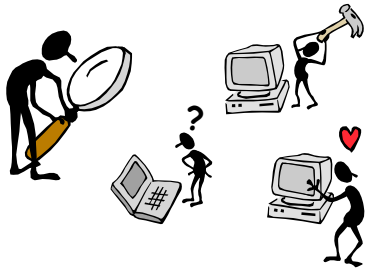
User-centered Design

Know Thy User

- Cognitive abilities
  - perception
  - physical manipulation
  - memory
- Organizational / job abilities
- Keep users involved throughout
  - developers working with target users
  - think of the world in users' terms (empathy)
  - understanding work process
  - not technology-centered / feature driven

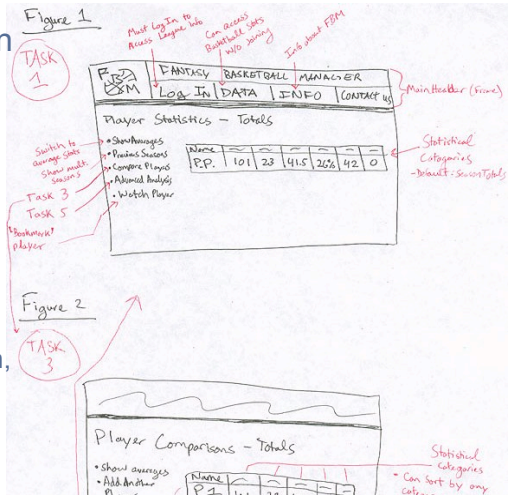
Task Analysis & Contextual Inquiry

- Observe existing work practices
- Create examples and scenarios of actual use
- "Try-out" new ideas before building software



Rapid Prototyping

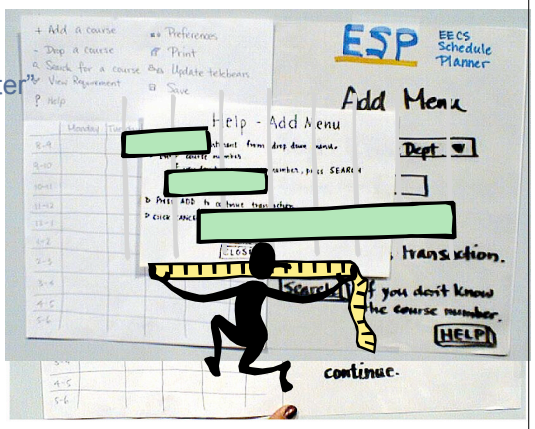
- Build a mock-up of design so you can quickly test
- Low fidelity techniques
  - paper sketches
  - cut, copy, paste
- Interactive prototyping tools
  - HTML, Visual Basic, HyperCard, Director, Flash, DENIM, etc.
- UI builders
  - Visual Studio .NET, JBuilder…



Fantasy Basketball

Evaluation

- Test with real users (participants)
  - w/ interactive prototype
  - low-fi with paper "computer"
- Build models
  - GOMS
- Low-cost techniques
  - expert evaluation
  - walkthroughs
  - online testing



ESP

Programming

- Toolkits
- UI Builders
- Event models
- Input / Output models
- etc.

Won't discuss these in this course

Goals of this Course

- Learn to design, prototype, & evaluate UIs
  - the needs & tasks of prospective users
  - cognitive/perceptual constraints that affect design
  - technology & techniques used to prototype UIs
  - techniques for evaluating a user interface design
  - importance of iterative design for usability
- Focus on both research and practice

Course Format

- Interactive lectures
  - Feel free to ask lots of questions
  - Tell us about your experiences or problems too
- Have fun & participate!

Schedule

1:30 –	**2:00**	Introduction & Course Overview
2:00 –	**2:15**	Introductions
2:15 –	**3:00**	Design Discovery
3:00 –	**3:10**	10 min break
3:10 –	**3:45**	Prototyping
3:45 –	**4:15**	Mental Models
4:15 –	**5:00**	Evaluation
5:00 –	**.**	Open Course Discussion

Introductions

1. Name & what your responsibilities are
2. One or more of the following:
   - What is your biggest design headache?
   - What is your biggest customer complaint?
   - What are you looking to get out of the course?

Further Reading
Introduction to HCI

- Books
  - *Designing the User Interface* by Ben Shneiderman
  - *The Design of Everyday Things* by Don Norman
  - *Task-Centered User Interface Design: A Practical Introduction* (online) by Lewis & Rieman
  - *Designing Web Usability* by Jakob Nielsen
- Articles
  - Is software too hard to use?, *ComputerWorld*, Aug. '99
  - *Interactions* magazine, http://www.acm.org/interactions/

Further Reading
Introduction to HCI

- Web Sites
  - useit.com
  - HCI index at http://degraaff.org/hci/
- Organizations
  - ACM SIGCHI, BayCHI, UPA, Stanford PCD Seminar

User Interface Design, Prototyping, and Evaluation

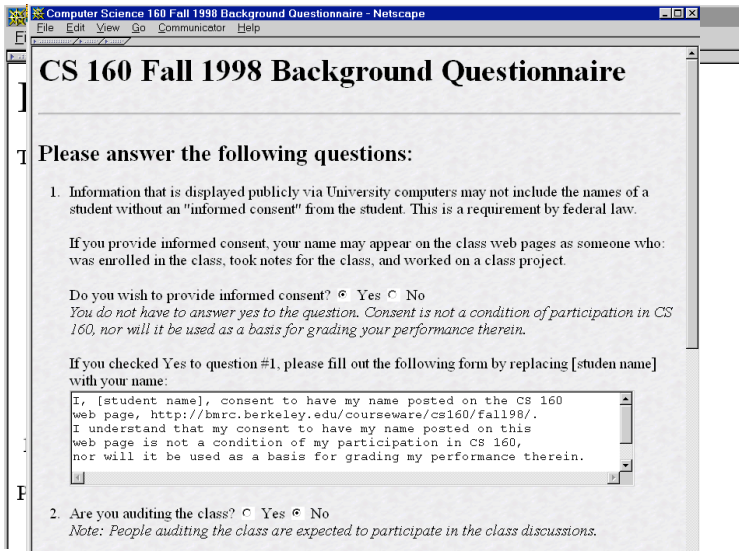
Design Discovery

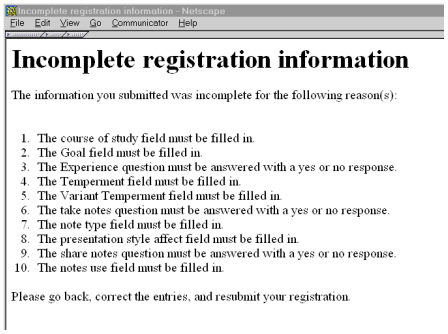
Jason I. Hong

July 6, 2005

Carnegie Mellon Human-Computer Interaction Institute

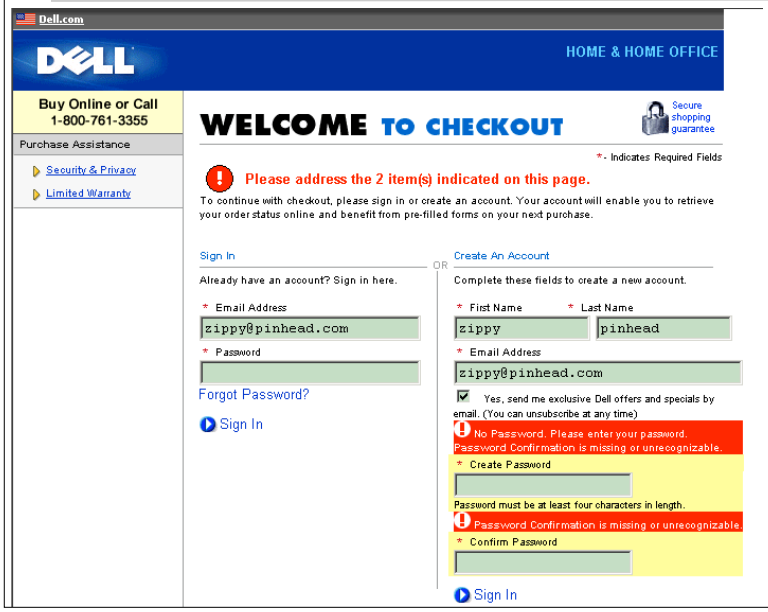
Interface Hall of Shame or Fame?





- Comments?

A Better Approach…



User Interface Design, Prototyping, and Evaluation

Design Discovery

Jason I. Hong

July 6, 2005

Carnegie Mellon Human-Computer Interaction Institute

Outline

- Understanding the user
- Task analysis
- Selecting & using tasks in design
- Contextual inquiry

"You Are Not the User"

- Easy to think of self as typical user
- Easy to make mistaken assumptions

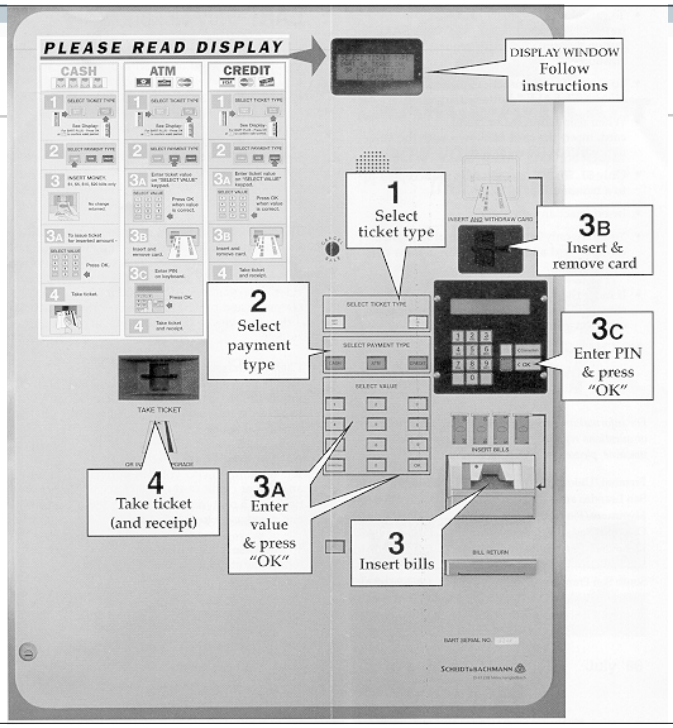
- People have:
  - Different backgrounds and experiences
  - Different needs and responsibilities
  - Different terminology
  - Different ways of looking at the world

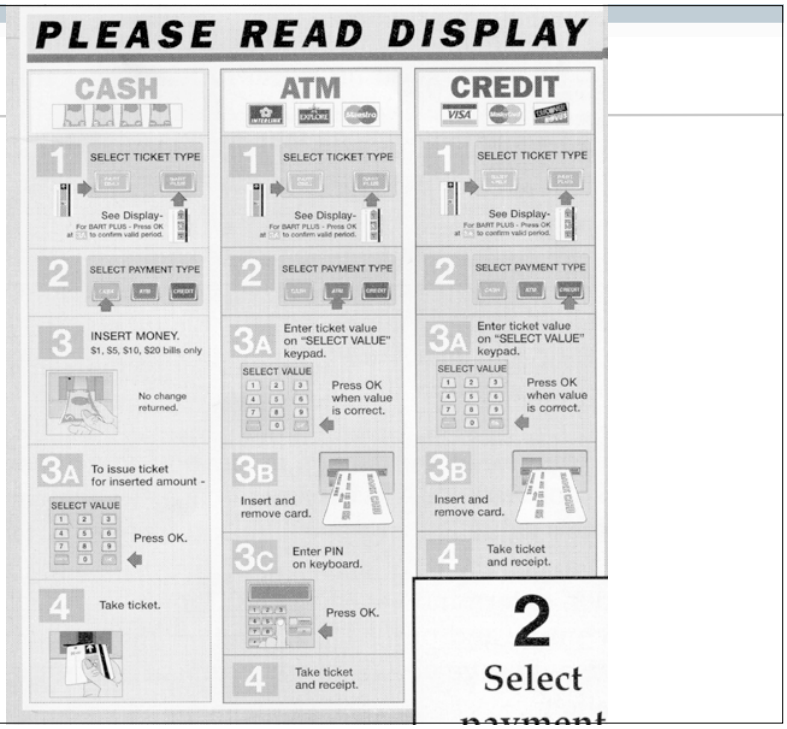
Understanding the User

- 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!

Example of Design Failure

- BART “Charge-a-Ticket” Machines
  - allow riders to buy BART tickets or add fare
  - takes ATM cards, credit cards, & cash





Example of Design Failure

- BART "Charge-a-Ticket" Machines
  - allow riders to buy BART tickets or add fare
  - takes ATM cards, credit cards, & cash
- Problems (?)

Lessons from the BART machine

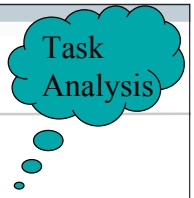
- Failure to create convenient machine
- Systems will fail if they:
  - do not do what the user needs
  - are inappropriate
  - Ex. poor sales, slow, errors, passive or active resistance
- Can't we just define "good" interfaces?
  - "good" has to be taken in context of users
    - might be acceptable for office work, not for play
    - infinite variety of tasks and users
  - guidelines are too vague to be generative
    - e.g., "give adequate feedback"
- How can we avoid similar results?
  - "What is required to perform the user's task?"

Task Analysis

- Find out:
  - who users are
  - what tasks they need to perform
- Observe existing work practices
- Create scenarios of actual use

- This lets us try new ideas *before* building software!
  - Get rid of problems early in the design process while they are still cheap to fix!

Task Analysis Questions



- 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?

Task Analysis Questions (cont.)

- 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
- Work habits and preferences
- Physical characteristics
  - height?




Who (BART)?

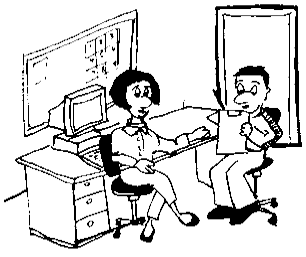
- Identity?
  - people who ride BART
    - business people, students, disabled, elderly, tourists
- Background?
  - may have an ATM or credit card
  - have used other fare machines before
- Skills?
  - may know how to put cards into ATM
  - know how to buy BART tickets

Who (BART cont.)?

- Work habits and preferences?
  - Some people: use BART 5 days a week
  - Others: first time use
- Physical characteristics?
  - varying heights –> don't make it too high or too low!

Talk to Potential Users

- Find some real users
- Talk to them
  - find out what they do
  - how would your system fit in
- Are they too busy?
  - buy their time
    - t-shirts, coffee mugs, etc.
  - find substitutes
    - medical students in training



What Tasks?

- Important for both automation and new functionality
- Relative importance of tasks?
- Observe users, see things from their perspective

- Example: on-line billing
  - small dentists office had billing automated
  - assistants were unhappy with new system
  - old forms contained hand-written margin notes
    - e.g., patient A's insurance takes longer than most, etc.

How are Tasks Learned?

- What does the user need to know?
- Do they need training?
  - academic
  - general knowledge / skills
  - special instruction / training

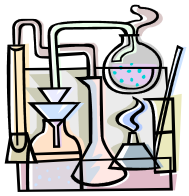
Where is the Task Performed?

- Office, laboratory, point of sale?
- Effects of environment on users?
- Users under stress?
- Confidentiality required?
- Wet, dirty, or slippery hands?
- Soft drinks?
- Lighting?
- Noise?

What is the Relationship Between Users & Data?

- Public data?
  - Open government records, public web sites
- Personal data?
  - Ex. health records, bank records
  - always accessed at same machine?
  - do users move between machines?
- Common data?
  - used concurrently?
  - passed sequentially between users?
- Remote access required?
- Access to data restricted?

What Other Tools Does the User Have?



- More than just compatibility
- How user works with collection of tools

- Example: automating lab data collection
  - how is data collected now?
  - by what instruments and manual procedures?
  - how is the information analyzed?
  - are the results transcribed for records or publication?
  - what media/forms are used and how are they handled?

How Do Users Communicate With Each Other?

- Who communicates with whom?
- About what?
- Follow lines of the organization? Against it?

How Often Do Users Perform the Tasks?

- Frequent users remember more details
- Infrequent users may need more help
  - even for simple operations
  - make these tasks possible to do
- Which function is performed
  - most frequently?
  - by which users?
  - optimize system for these tasks will improve perception of good performance

What are the Time Constraints on the Task?

- What functions will users be in a hurry for?
- Which can wait?
- Is there a timing relationship between tasks?

What Happens When Things Go Wrong?

- How do people deal with
  - task-related errors?
  - practical difficulties?
  - catastrophes?
- Is there a backup strategy?

Involve Users to Answer Task Analysis Questions

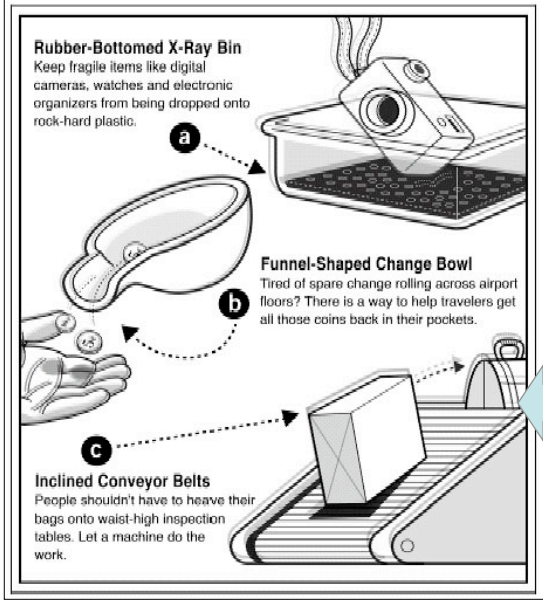
- Users help designers learn
  - what is involved in their jobs
  - what tools they use
  - i.e., what they do
- Developers reveal technical capabilities
  - builds rapport & an idea of what is possible
  - user's can comment on whether ideas make sense
- How do we do this?
  - observe & interview prospective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

A Better Ticket Machine

New BART Machine

Hong Kong MTR System

1 Minute Break



- Good design matters in all areas of our lives
- The little things really do matter
- A designer's proposed changes to airport screenings

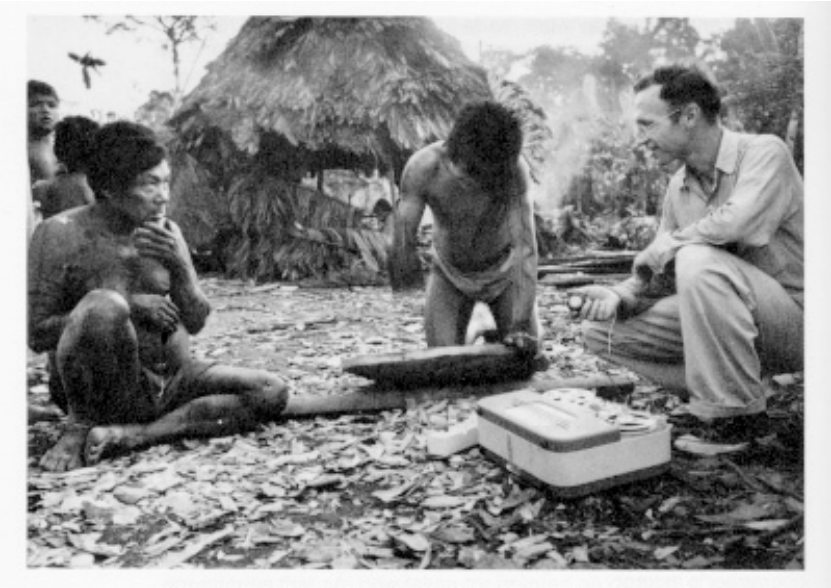
Design from Data

- So far, haven't involved real users yet
  - Task analysis involves best guesses about users
  - Need a way of confirming your task analysis!
- Heated arguments with others on design team
  - Over what tools, skills, and knowledge users have
  - These tend to generate lots of heat, little light
- Key to solving this: design from data!
  - Go out to real users and get real data from them

Can't we just ask users what they want?

- Not familiar with what is possible with technology
- Not familiar with design constraints
  - Budget, legacy code, time, etc
- Not familiar with good design
- Not familiar with security and privacy
- Sometimes users don't know what they want
  - Ex. Remote controls
- Contextual inquiry is an important method for understanding users' needs
  - Later on in this course, fulfilling those needs through designs

“Classic” Ethnography



Ethnography and HCI

- More recently, growing interest in applying ethnography to system design
  - e.g. Hughes, et al. air traffic controllers study

Contextual Inquiry

- A quick and dirty form of ethnography
  - A way of understanding users' needs and work practices
- Uses a Master / Apprentice model for 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 - 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?"

Principles - 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 - Focus

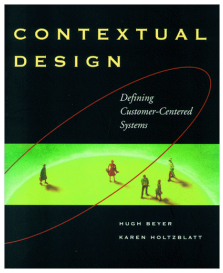
- Interviewer needs data about specific kind of work
  - “steer” conversation to stay on useful topics
- Respect triggers (flags to change focus)
  - shift of attention (someone walks in)
  - surprises (you know it is “wrong”)

Users: Unique or One of Many?

"Take the attitude that nothing any person does is done for no reason; if you think it's for no reason, you don't yet understand the point of view from which it makes sense.

Take the attitude that nothing any person does is unique to them, it always represents an important class of customers whose needs will not be met if you don't figure out what's going on."

(p. 63, *Contextual Design*)



Thoughts on Interviews

- 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 Users 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
  - (even if you helped create the system)
- Be careful not to judge participants

- REMEMBER: your goal is to make the system easy to use for your intended users

Using the Data You Learn

- Figure out what is important
- Affinity diagramming
  - group info & find relations between groups
  - Post-Its on large surfaces
    - immersive
    - persistent
    - brainstorming
  - also used for creating web info architecture

Using the Data You Learn

- Say who the users are (use personas or profiles)

Example Persona

Name:	Patricia
Age:	31
Occupation:	Sales Manager, IKEA Store
Hobbies:	Painting Fitness/biking Taking son Devon to the park
Likes:	Emailing friends & family Surprises for her husband Talking on cell phone with friends Top 40 radio stations Eating Thai food Going to sleep late
Dislikes:	Slow service at checkout lines Smokers

Using the Data You Learn

- Say who the users are (use personas or profiles)
  - personas do not have to be a real person, but should be based on real facts and details
  - design can really differ depending on who
  - name names (allows getting more info later)
  - characteristics of the users (job, expertise, etc.)
- Might have one persona for each class of users
  - helps the design team think in terms of the users

Selecting Tasks

- Choose real tasks users face
  - collect any necessary materials
- Should provide reasonable coverage
  - compare check list of features to selected tasks
- Have a mixture of simple & complex tasks
  - easy tasks (common or introductory)
  - moderate tasks
  - difficult tasks (infrequent or for power users)
- Good tasks are fundamental to good usability
  - does your system support key tasks desired by users?
  - how well does your system support these?

What Should Tasks Look Like?

- Say what the user wants to do, but not how
  - allows comparing different design alternatives
- Being specific also forces us to fill out description with relevant details
  - example: file browser story
  - makes sure our tasks are realistic!
- Some tasks should describe a complete story
  - example: phone-in bank functions
  - forces us to consider how features work together

Using Tasks in Design

- Write up a description of tasks
  - formally or informally
  - run by users and rest of the design team
    - do they make sense? realistic?
  - get more information where needed

Manny is in the city at a club and would like to call his girlfriend, Sherry, to see when she will be arriving at the club. She called from a friend's house while he was on BART, so he couldn't answer the phone. He would like to check his missed calls and find the number so that he can call her back.

Using Tasks in Design (contd.)

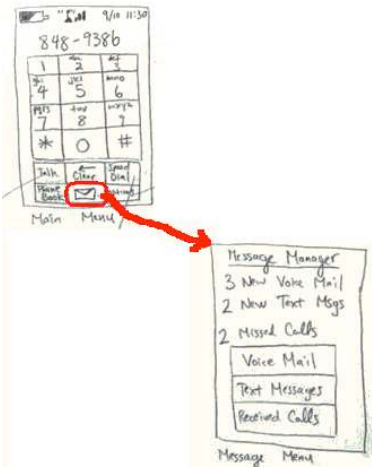
- Okay to have some freeform tasks
  - Ex: “purchase tickets for a movie you want to see”
    - navigation, reviews, shopping cart, etc
  - specific tasks good for understanding usability, freeform tasks good for understanding usefulness

Using Tasks in Design (cont.)

- 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

Scenarios (cont.)

- Scenarios are *design specific*, tasks aren't
- Scenarios force us to
  - show how various features will work together
  - settle design arguments by seeing (and testing!) examples
- Show users storyboards
  - get feedback



Caveats of User-Centered Design Techniques

- Politics
  - "agents of change" can cause controversy
  - get a sense of organization & bond w/ interviewee
  - important to get buy-in from all those involved
- 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

Summary

- Know thy user & involve them in design
  - answer questions before designing
    - who, what, where, when, how often?
    - users & data?, other tools? when things go wrong?
- Selecting tasks
  - real tasks with reasonable functionality coverage
  - complete, specific tasks of what user wants to do
- Contextual inquiry
  - way to answer the task analysis questions
  - design from real data
  - interview & observe real users
  - use the master-apprentice model to get them to teach you

Further Reading

Task Analysis, Contextual Inquiry, & Personas

- Books
  - *User and Task Analysis for Interface Design* by Joann T. Hackos, Janice C. Redish
  - *Contextual Design* by Hugh Beyer & Karen Holtzblatt
  - *The Inmates are Running the Asylum* by Alan Cooper
- Articles
  - Beyer, Hugh, and Holtzblatt, Karen, "Apprenticing with the Customer: A Collaborative Approach to Requirements Definition," *Communications of the ACM*, May 1995.
- Web Sites
  - Beyer, Hugh, "Getting Started with Contextual Techniques"
    - http://www.incent.com/connection.indx/techniques.html

User Interface Design, Prototyping, and Evaluation

Prototyping

Jason I. Hong

July 6, 2005

Carnegie Mellon Human-Computer Interaction Institute

Hall of Fame or Shame?

- Password dialog in Eudora Pro for Mac

- Password dialog in Eudora Pro for Mac

User Interface Design, Prototyping, and Evaluation

Prototyping

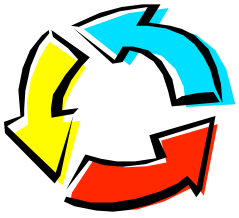
Jason I. Hong
July 6, 2005

Outline

- Low-fi prototyping
- Wizard of Oz technique
- Informal UI prototyping tools
- Hi-fi prototyping
- What prototyping tools lack

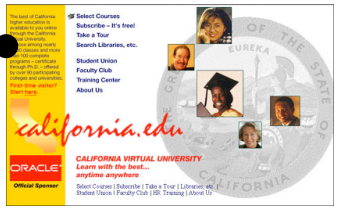
Why Do We Prototype?

- Quickly experiment with alternative designs
- Get feedback on our design faster
  - fix problems before code is written
  - saves time and money
- Keep the design centered on the user
  - must test & observe ideas with users

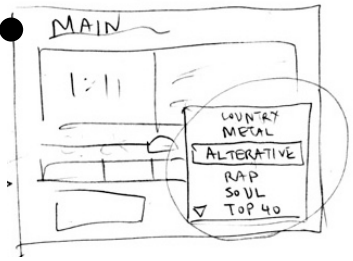


Fidelity in Prototyping

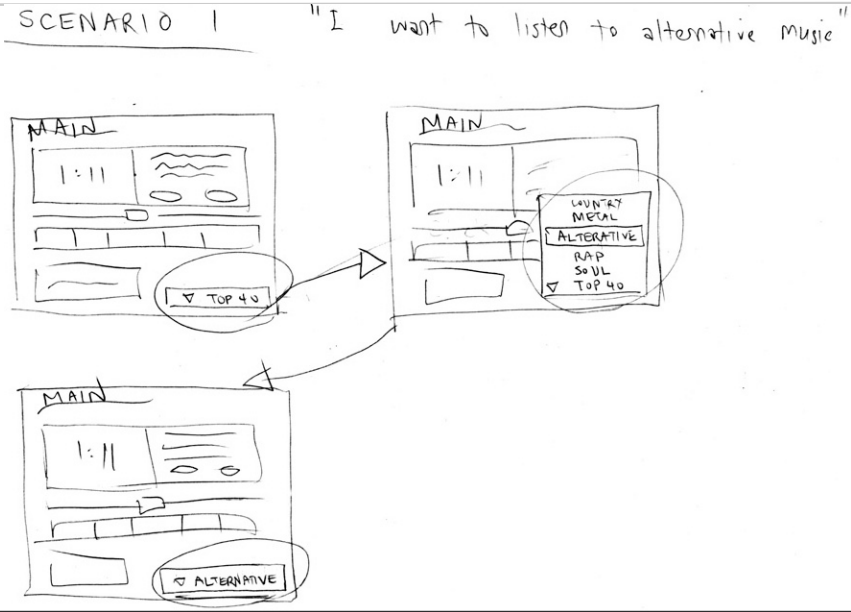
- *Fidelity* refers to level of detail
- High fidelity
  - prototype looks like the final product



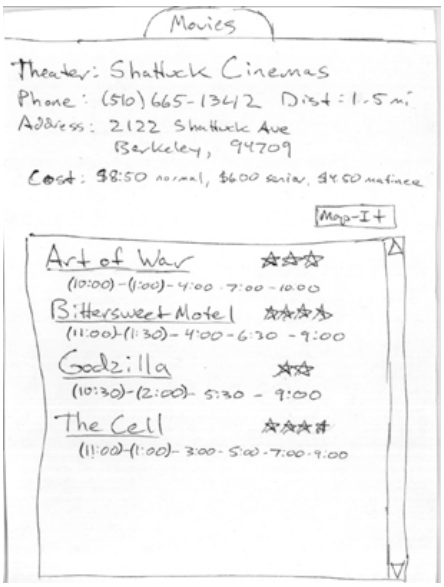
- Low fidelity
  - artist's rendition with many details missing

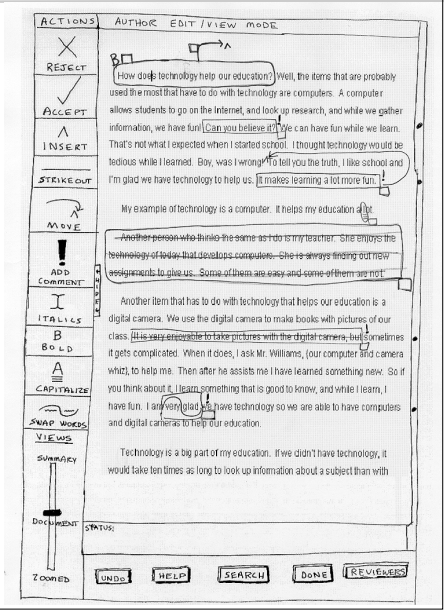


Low-fi Sketches & Storyboards

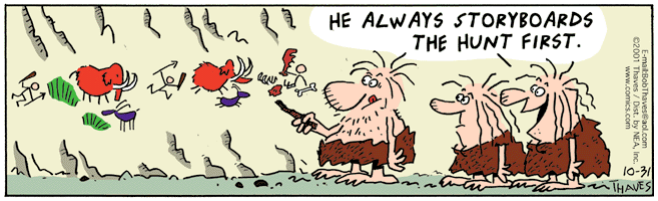


Low-fi Sketches & Storyboards

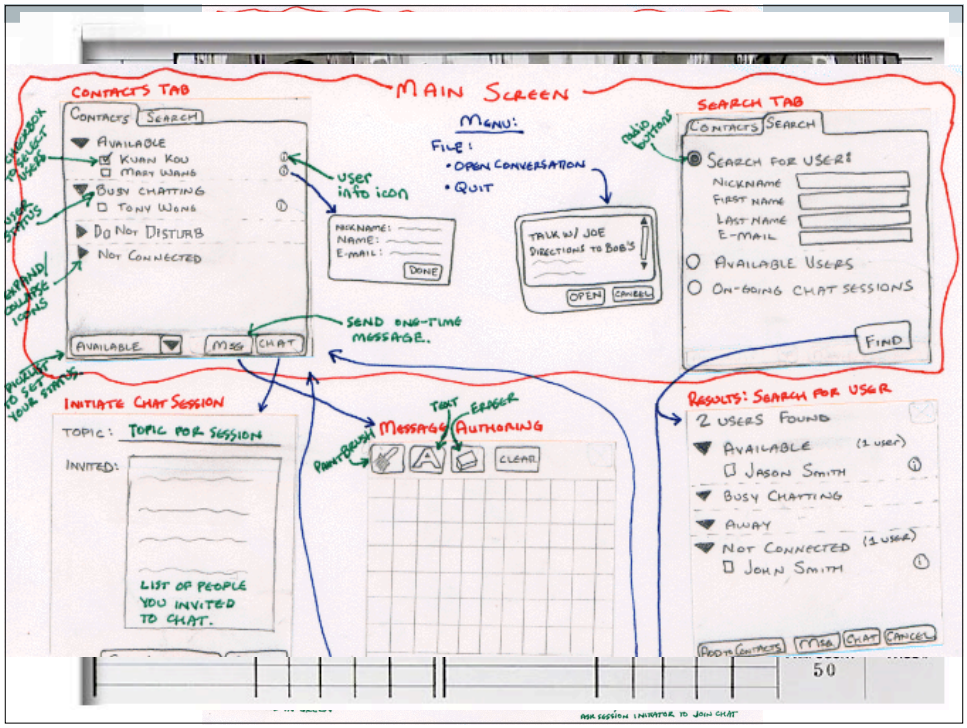




Low-fi Sketches & Storyboards



- Where do storyboards come from?
- Give you a "script" of important events
  - leave out the details
  - concentrate on the important interactions

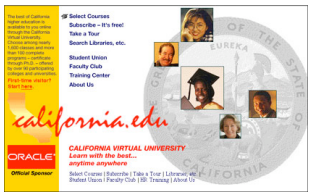


Why Use Low-fi Prototypes?

- Traditional methods take too long
  - sketches -> **build prototype** -> evaluate -> iterate
  - don't want to program for weeks or months before feedback
- *Simulate* the prototype
  - sketches -> evaluate -> iterate
  - sketches act as prototypes
    - designer "plays computer"
    - other design team members observe & record
- Kindergarten implementation skills
  - allows non-programmers to participate
  - helps make sure everyone on the team is together

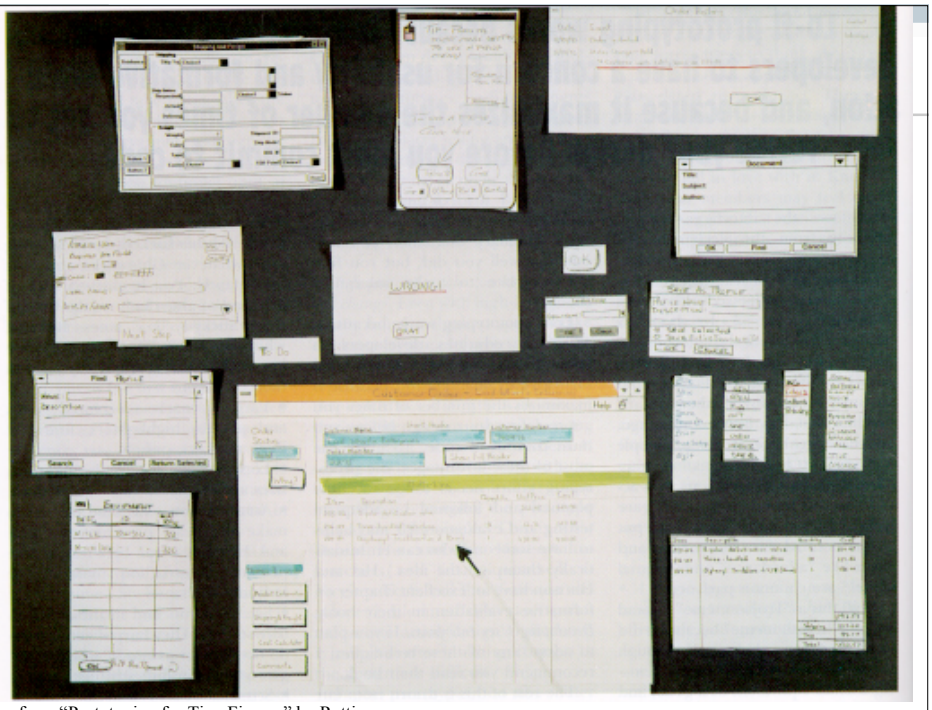
Hi-fi Prototypes Affect Feedback

- Perceptions of the tester/reviewer
  - representation communicates “finished”
    - comments focus on color, fonts, & alignment
- Time
  - hi-fi tools encourage precision
    - specifying details takes more time
- Creativity
  - lose track of the big picture

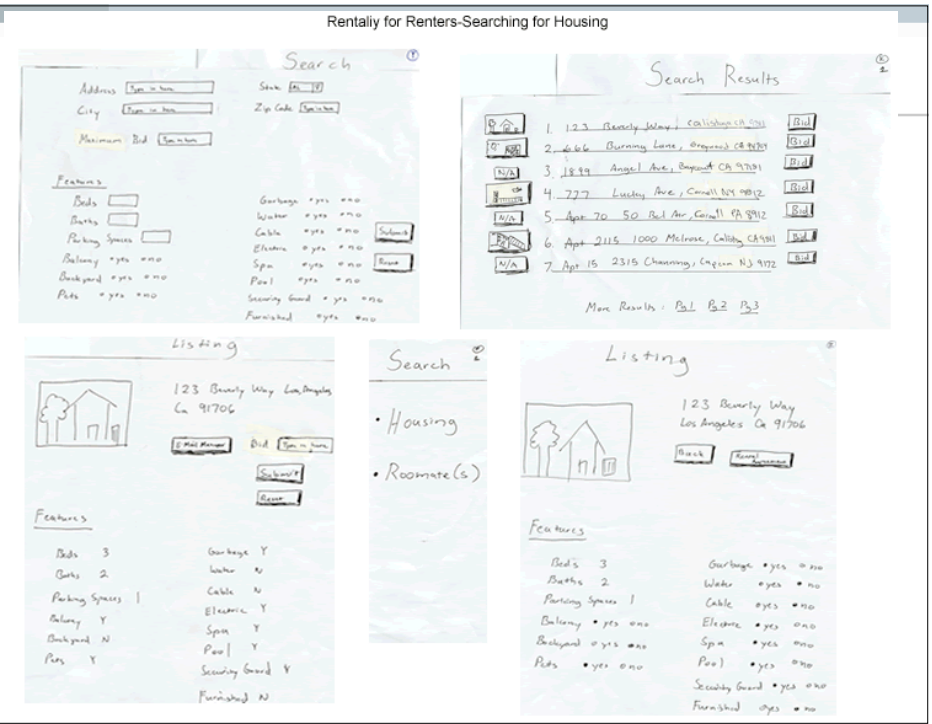


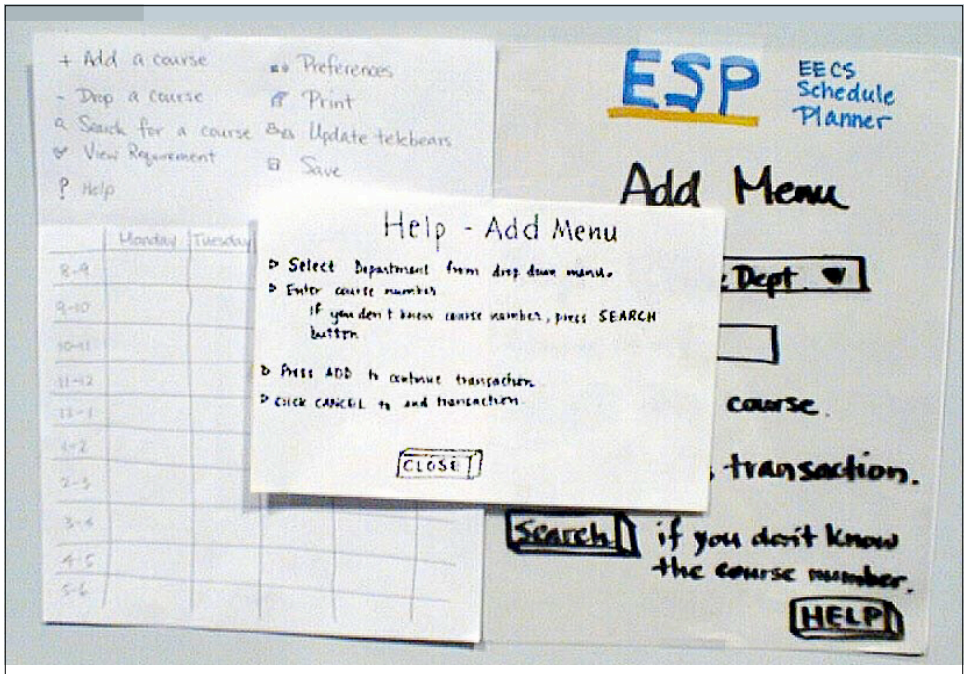
The Basic Materials

- Large, heavy, white paper (11 x 17)
- 5x8 in. index cards
- Post-its
- Tape, stick glue, correction tape
- Pens & markers (many colors & sizes)
- Overhead transparencies
- Scissors, X-acto knives, etc.



from “Prototyping for Tiny Fingers” by Rettig





ESP

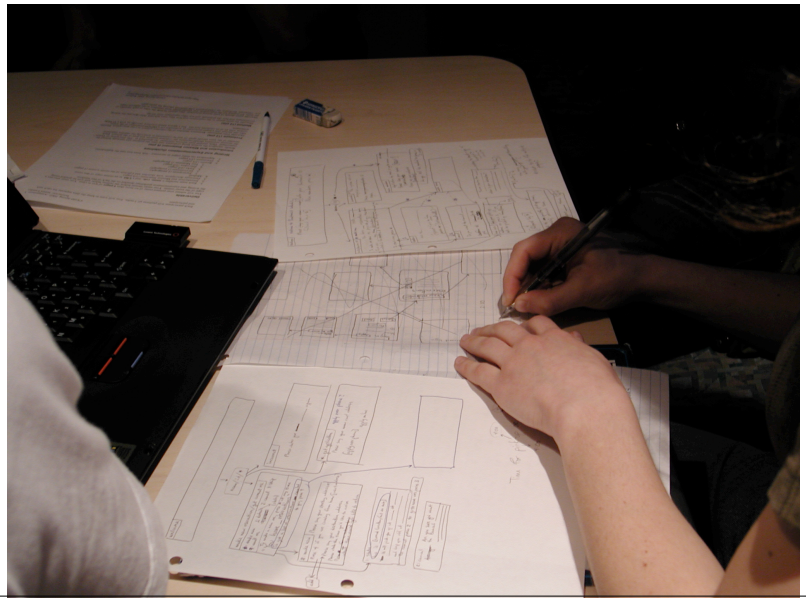
Constructing the Paper Prototype

- Set a deadline
  - a few hours or 1-2 days
  - don't think for too long - build it!
- Draw a window frame on large paper
- Put different screen regions on cards
  - anything that moves, changes, appears/disappears
- Ready response for any user action
  - e.g., have those pull-down menus already made
- Use photocopier to make many versions

Constructing the Model

Constructing the Model

Constructing the Model



Preparing for a Test

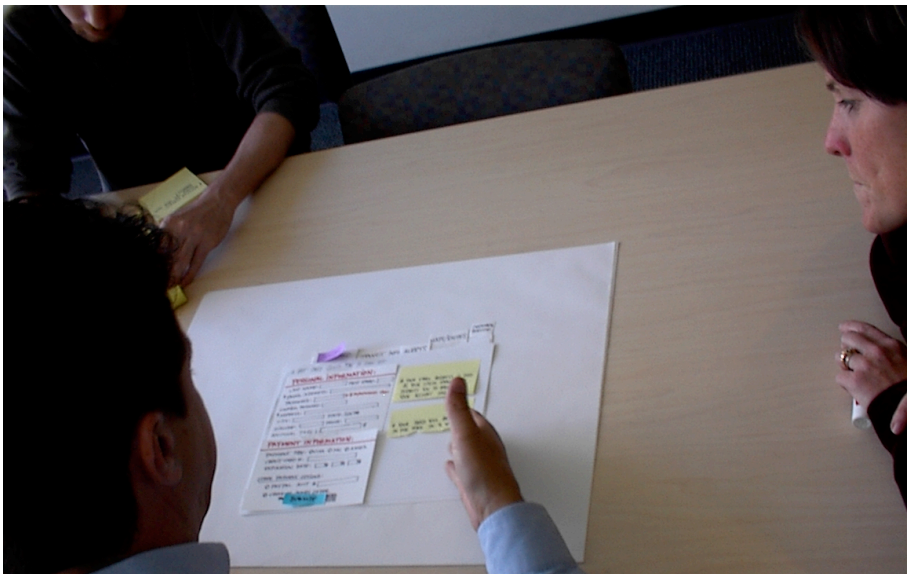
- Select your users
  - understand background of intended users
  - use a questionnaire to get the people you need
  - minimize use friends or family
- Prepare scenarios that are
  - typical of the product during actual use
  - make prototype support these (small, yet broad)
- Practice to avoid “bugs”

Conducting a Test

- Four testers (minimum)
  - greeter – puts users at ease & gets data
  - facilitator – only team member who speaks
    - gives instructions & encourages thoughts, opinions
  - computer – knows application logic & controls it
    - always simulates the response, w/o explanation
  - observers – take notes & recommendations
- Typical session is 1 hour
  - preparation, the test, debriefing

Conducting a Test

Conducting a Test



Co

Conducting a Test

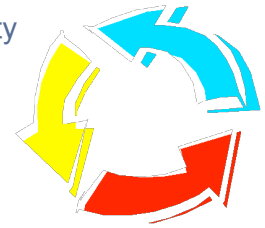
- Have good tasks
  - Well-defined and good tasks are key to good usability
  - Present tasks to users on sheet of paper
- Don't tell them too much
  - You want them to "think-aloud"
  - Explain the basic concept of the UI, but not too much
    - You want to see if they can figure it out
- Do a debriefing at the end
  - Were tasks realistic?
  - What parts made sense? Confusing?
  - Any features missing?

Evaluating Results

- Sort & prioritize observations
  - what was important?
  - lots of problems in the same area?
- Create a written report on findings
  - gives agenda for meeting on design changes
- Make changes & iterate

Advantages of Low-fi Prototyping

- Takes only a few hours
  - no expensive equipment needed
- Can test multiple alternatives
  - fast iterations
    - number of iterations is tied to final quality
- Almost all interaction can be faked



Wizard of Oz Technique

- Faking the interaction. Comes from?

Wizard of Oz Technique

- Faking the interaction. Comes from?
  - the film “The Wizard of OZ”
    - “the man behind the curtain”
- Long tradition in computer industry
  - e.g., prototype of a PC w/ a VAX behind the curtain
- Much more important for hard to implement features
  - speech & handwriting recognition

Closing Thoughts

- Don’t over-think
  - Just build it!
  - Can do several iterations in days
  - Best bang for buck
- Don’t skip this step
  - Time, budget, etc
  - Better to fix bugs now than later
  - Makes sure you have the right features
    - Easier to fix a bug now than after a release
- Get everyone involved
  - Good for non-programmers

Summary

- Low-fi testing allows us to quickly iterate
  - get feedback from users & change right away
- Informal prototyping tools bridge the gap between paper & high-fi tools
- High-fi tools good for testing more developed UI ideas
  - generally ignore the "insides" of application

Further Reading
Prototyping

- Books
  - Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces, by Carolyn Snyder, Morgan Kaufmann, 2003
- Articles
  - "Prototyping for Tiny Fingers" by Marc Rettig, in *Communications of the ACM*, 1994
  - "Using Paper Prototypes to Manage Risk" by Carolyn Snyder, http://world.std.com/~uieweb/paper.htm
  - "The Perils of Prototyping" by Alan Cooper, http://www.chi-sa.org.za/Documents/articles/perils.htm
- Web Sites
  - Group for User Interface Research, for DENIM & SUEDE downloads, http://guir.berkeley.edu
  - InfoDesign Toolkit, http://www.infodesign.com.au

User Interface Design, Prototyping, and Evaluation

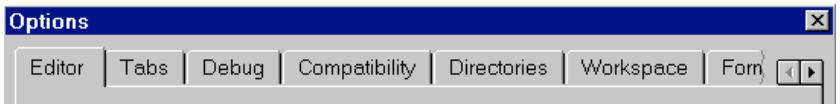
Conceptual Models & Interface Metaphors

Jason I. Hong

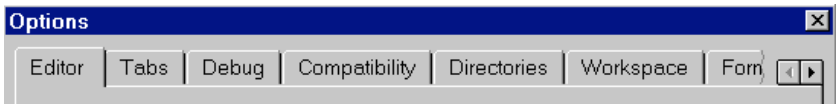
July 6, 2005

Carnegie Mellon Human-Computer Interaction Institute

Interface Hall of Fame or Shame?



- Tabbed dialog for setting options in MS Web Studio
  - more tabs than space to display them
- Clicking on the right arrow once gives:



- Tabbed dialog for setting options in MS Web Studio
  - more tabs than space to display them
- Clicking on the right arrow once gives:

User Interface Design, Prototyping, and Evaluation

Conceptual Models & Interface Metaphors

Jason I. Hong

July 6, 2005

Carnegie Mellon Human-Computer Interaction Institute

Outline

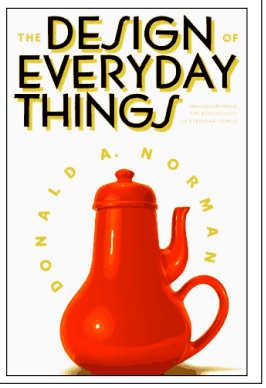
- *Design of Everyday Things*
- Conceptual models
- Interface metaphors

Design of Everyday Things

- By Don Norman (UCSD, Apple, HP, NN Group)
- Design of everyday objects illustrates problems faced by designers of systems

- Explains conceptual models
  - doors, washing machines, digital watches, telephones, ...
- Resulting design guides

–> Highly recommend this book



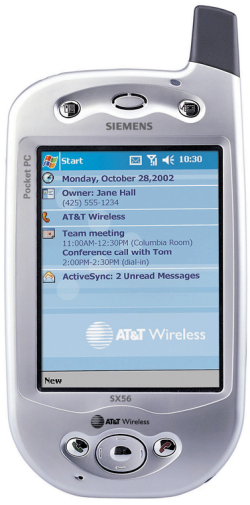
Conceptual Models

- Mental representation of how object works & how interface controls affect it
- People may have preconceived models that are hard to change
  - (4 + 5) vs. (4 5 +)
  - dragging to trash?
    - delete file but eject disk
- Interface must communicate model
  - visually
  - online help and documentation can help, but shouldn't be necessary

Affordances as Perceptual Clues

- Well-designed objects have *affordances*
  - clues to their operation
  - often visual, but not always (e.g., speech)

Affordances as Perceptual Clues

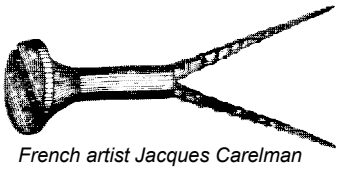


Siemens Pocket PC Phone
Pen input, no keypad

Handspring Treo
Pen input/keypad input

Affordances as Perceptual Clues

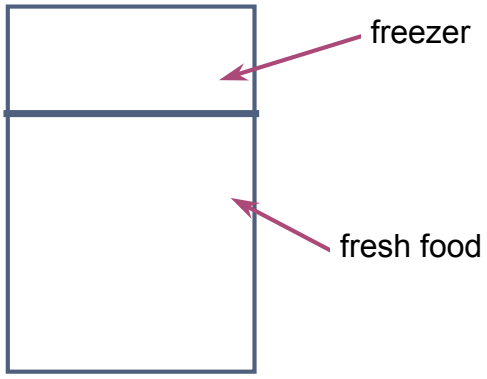
- Poorly-designed objects
  - no clues or misleading clues



French artist Jacques Carelman

Crazy design for a screw punch!

Refrigerator



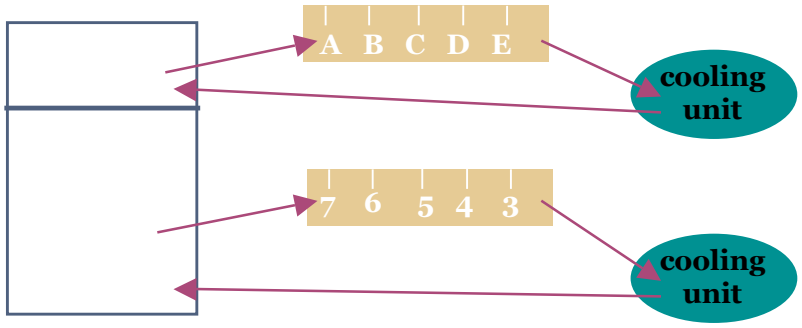
Problem: freezer too cold, but fresh food just right

Refrigerator Controls

Normal Settings	**C** and **5**
Colder Fresh Food	**C** and **6-7**
Coldest Fresh Food	**B** and **8-9**
Colder Freezer	**D** and **7-8**
Warmer Fresh Food	**C** and **4-1**
OFF *(both)*	**0**

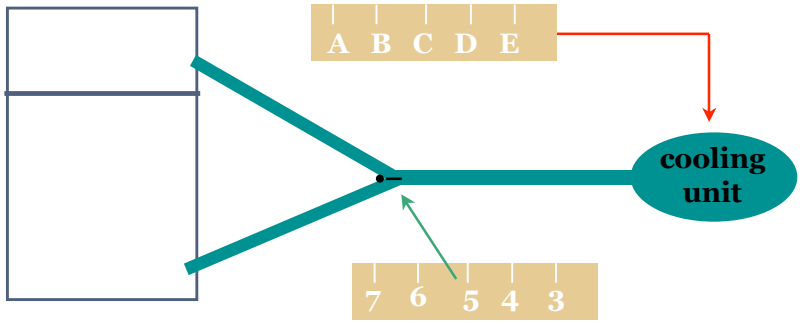
What is your conceptual model?

A Common Conceptual Model



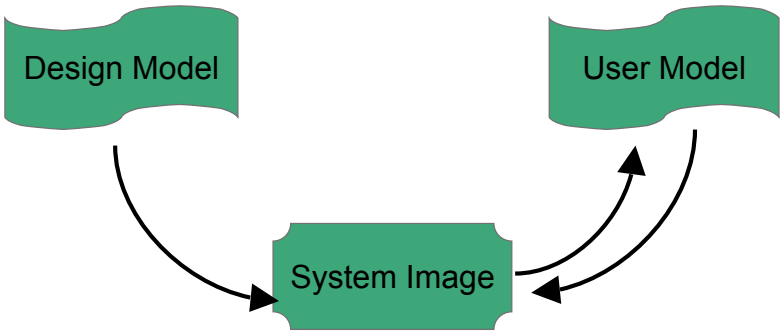
independent controls

Actual Conceptual Model



- Now can you fix the problem?
- Possible solutions
  - make controls map to user's model
  - make controls map to actual system

Design Model & User Model

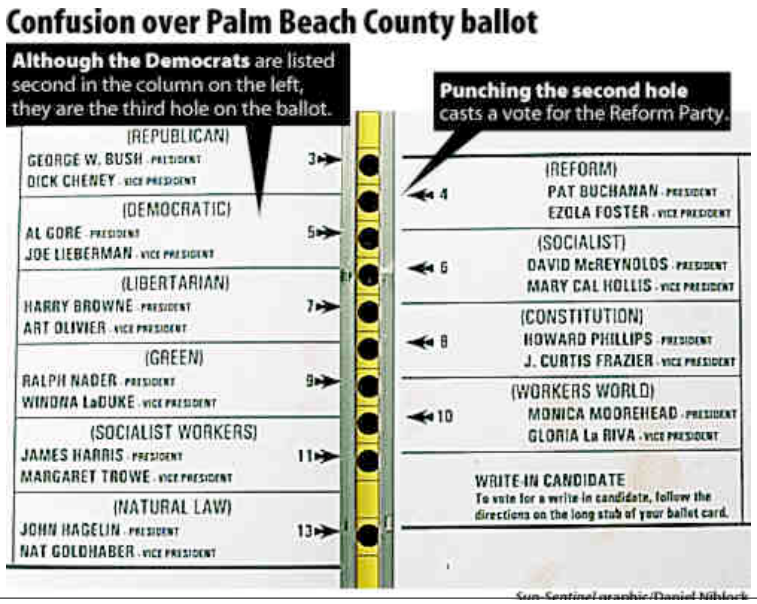


- Users get model from experience & usage
  - through system image
- What if the two models don't match?

Conceptual Model Mismatch

- Mismatch between designer's & user's conceptual model leads to…
  - Slow performance
  - Errors
    - And inability to recover
  - Frustration
  - ...

Notorious Example



A Personal Example

Design Guides

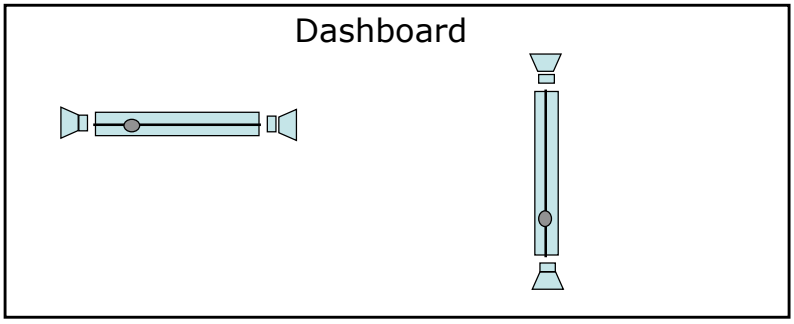
- Provide good conceptual model
  - user wants to understand how UI controls impact object
- Make things visible
  - if object has function, interface should show it
- Map interface controls to user's model
  - infix vs. postfix calculator -- whose model?
- Provide feedback
  - what you see is what you get!

Make Things Visible

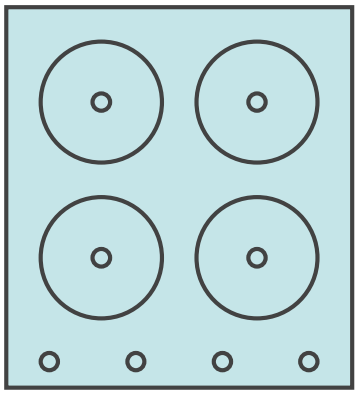
- Refrigerator (?)
  - make the A..E dial something about percentage of cooling between the two compartments?
- Controls available on watch w/ 3 buttons?
  - too many and they are not visible!
- Compare to controls on simple car radio
  - #controls = #functions
  - controls are labeled (?) and grouped together

Map Interface Controls

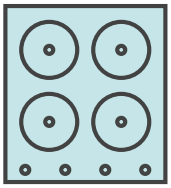
- Control should mirror real-world
- Which is better for dashboard speaker front / back control?



Map Interface Controls



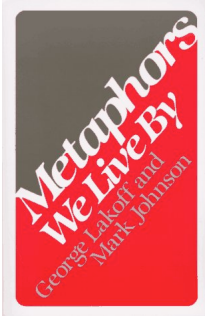
Map Interface Controls



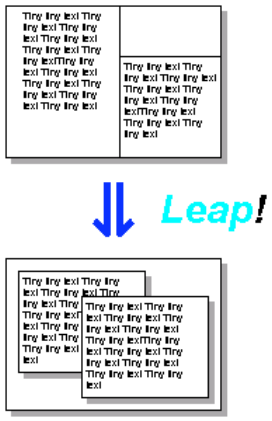
Metaphor

- Definition ?
  - "The transference of the relation between one set of objects to another set for the purpose of brief explanation."

- Lakoff & Johnson, *Metaphors We Live By*
  - "...the way we think, what we experience, and what we do every day is very much a matter of metaphor."
  - in our language & thinking - "argument is war"
    - he attacked every weak point
      ... criticisms right on target
      ... if you use that strategy
- We can use metaphors to leverage existing conceptual models



Desktop Metaphor



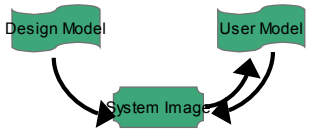
- Suggests a conceptual model
  - Not really an attempt to simulate a real desktop
  - Leverages existing knowledge about files, folders, trash
  - A way to explain why some windows seemed blocked

Example Metaphors

- Global metaphors
  - personal assistant, wallet, clothing, pens, cards, telephone, eyeglasses
- Data & function
  - rolodex, to-do list, calendar, applications documents, find, assist
- Collections
  - drawers, files, books, newspapers, photo albums

Summary

- Conceptual models
  - mental representation of how the object works & how interface controls effect it
- Design model should equal user model
  - mismatches lead to errors
  - know the user's likely conceptual model
- Design guides
  - make things visible
  - map interface controls to user's model
  - provide feedback



Further Reading

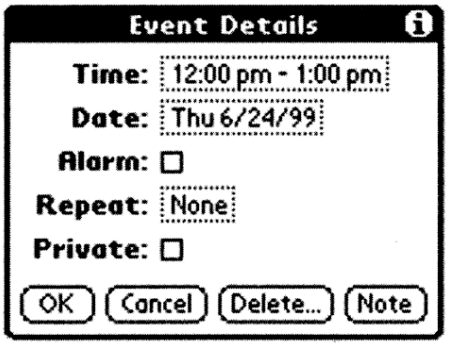
- *Design of Everyday Things*, Donald Norman
- Design as Practiced, Donald Norman
  - Talks about failure to make changes to Macintosh
  - http://www.jnd.org/dn.mss/Design_as_Practiced.html
- Computing the Case Against User Interface Consistency, Jonathan Grudin
  - Talks about why interfaces should not always be consistent
  - http://www1.ics.uci.edu/~grudin/Papers/CACM89/CACM89.html

Is Consistent Always Better?

- PDA example: should “add appointment” and “delete appointment” be in the same place?
- Add is common, but delete is not

Is Consistent Always Better?

Early Palm design



Streamlined design

Is Consistent Always Better?

- Interfaces should be consistent in a *meaningful* way
  - Eating knives, cutting knives, Swiss army
- Some types of consistency
  - Consistent internally
    - Ex. Same terminology and layout throughout
  - Consistent with other apps
    - Ex. Works like MSWord, uses keyboard conventions
    - Design patterns
  - Consistent with physical world

User Interface Design, Prototyping, and Evaluation

Evaluation

Jason I. Hong
July 6, 2005

Evaluation Outline

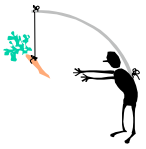
- Lab-based user testing
- Discount usability
- Remote / Online user testing
- Action analysis
- Heuristic evaluation

Why do User Testing?

- Can't tell how good UI is until people actually use it
- Other methods are based on evaluators who
  - may know too much
  - may not know enough (about tasks, etc.)
- Hard to predict what real users will do

Choosing Participants

- Representative of target users
  - job-specific vocab / knowledge
  - tasks
- Approximate if needed
  - system intended for doctors
    - get medical students
  - system intended for engineers
    - get engineering students
- Use incentives to get participants

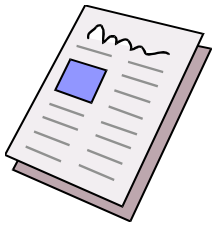


Ethical Considerations

- Sometimes tests can be distressing
  - users have left in tears
- You have a responsibility to alleviate
  - make voluntary with informed consent
  - avoid pressure to participate
  - let them know they can stop at any time
  - stress that you are testing the system, not them
  - make collected data as anonymous as possible
- Often must get human subjects approval

User Test Proposal

- A report that contains
  - objective
  - description of system being testing
  - task environment & materials
  - participants
  - methodology
  - tasks
  - test measures
- Get approved & then reuse for final report
- Seems tedious, but writing this will help "debug" your test

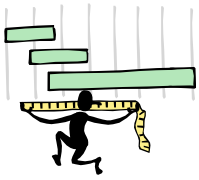


Selecting Tasks

- Should reflect what real tasks will be like
- Tasks from analysis & design can be used
  - may need to shorten if
    - they take too long
    - require background that test user won't have
- Try not to train unless that will happen in real deployment
- Avoid bending tasks in direction of what your design best supports
- Don't choose tasks that are too fragmented
  - e.g., phone-in bank test

Deciding on Data to Collect

- Two types of data
  - process data
    - observations of what users are doing & thinking
  - bottom-line data
    - summary of what happened (time, errors, success)
    - i.e., the dependent variables

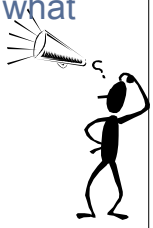


Which Type of Data to Collect?

- Focus on process data first
  - gives good overview of where problems are
- Bottom-line doesn't tell you where to fix
  - just says: "too slow", "too many errors", etc.
- Hard to get reliable bottom-line results
  - need many users for statistical significance

The "Thinking Aloud" Method

- Need to know what users are thinking, not just what they are doing
- Ask users to talk while performing tasks
  - tell us what they are thinking
  - tell us what they are trying to do
  - tell us questions that arise as they work
  - tell us things they read
- Make a recording or take good notes
  - make sure you can tell what they were doing



Thinking Aloud (cont.)

- Prompt the user to keep talking
  - "tell me what you are thinking"
- Only help on things you have pre-decided
  - keep track of anything you do give help on
- Recording
  - use a digital watch/clock
  - take notes, plus if possible
    - record audio & video (or even event logs)

Video of a Test Session

Using the Test Results

- Summarize the data
  - make a list of all critical incidents (CI)
    - positive & negative
  - include references back to original data
  - try to judge why each difficulty occurred
- What does data tell you?
  - UI work the way you thought it would?
    - users take approaches you expected?
  - something missing?

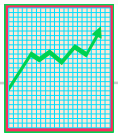
Using the Results (cont.)

- Update task analysis & rethink design
  - rate severity & ease of fixing CIs
  - fix both severe problems & make the easy fixes
- Will thinking aloud give the right answers?
  - not always
  - if you ask a question, people will always give an answer, even it is has nothing to do with facts
    - panty hose example
  - try to avoid specific questions

Measuring Bottom-Line Usability

- Situations in which numbers are useful
  - time requirements for task completion
  - successful task completion
  - compare two designs on speed or # of errors
- Ease of measurement
  - time is easy to record
  - error or successful completion is harder
    - define in advance what these mean
- Do not combine with thinking-aloud. Why?
  - talking can affect speed & accuracy

Analyzing the Numbers



- Example: trying to get task time <=30 min.
  - test gives: 20, 15, 40, 90, 10, 5
  - mean (average) = 30
  - median (middle) = 17.5
  - looks good!
- Wrong answer, not certain of anything!
- Factors contributing to our uncertainty
  - small number of test users (n = 6)
  - results are very variable (standard deviation = 32)
    - std. dev. measures dispersal from the mean

Analyzing the Numbers (cont.)

- This is what statistics is for
- Crank through the procedures and you find
  - 95% certain that typical value is between 5 & 55

Analyzing the Numbers (cont.)

Web Usability Test Results						
Participant #		**Time (minutes)**				
1		20				
2		15				
3		40				
4		90				
5		10				
6		5				
	number of participants	**6**				
	mean	**30.0**				
	median	**17.5**				
	std dev	**31.8**				
	standard error of the mean	**= stddev / sqrt (#samples)**			**13.0**	
	typical values will be mean +/- 2*standard error			**--> 4 to 56!**		
	what is plausible? = confidence (alpha=5%					

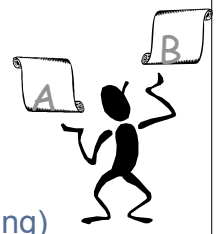
Analyzing the Numbers (cont.)

- This is what statistics is for
- Crank through the procedures and you find
  - 95% certain that typical value is between 5 & 55
- Usability test data is quite variable
  - need lots to get good estimates of typical values
  - 4 times as many tests will only narrow range by 2x
    - breadth of range depends on sqrt of # of test users
  - this is when online methods become useful
    - easy to test w/ large numbers of users

Measuring User Preference

- How much users like or dislike the system
  - can ask them to rate on a scale of 1 to 10
  - or have them choose among statements
    - "best UI I've ever…", "better than average"…
  - hard to be sure what data will mean
    - novelty of UI, feelings, not realistic setting …
- If many give you low ratings –> trouble
- Can get some useful data by asking
  - what they liked, disliked, where they had trouble, best part, worst part, etc. (redundant questions are OK)

Comparing Two Alternatives



- *Between groups* experiment
  - two groups of test users
  - each group uses only 1 of the systems
- *Within groups* experiment
  - one group of test users
    - each person uses both systems
    - can't use the same tasks or order (learning)
  - best for low-level interaction techniques
- Between groups requires many more participants than within groups
- See if differences are statistically significant
  - assumes normal distribution & same std. dev.

Experimental Details

- Order of tasks
  - choose one simple order (simple –> complex)
    - unless doing within groups experiment
- Training
  - depends on how real system will be used
- What if someone doesn't finish
  - assign very large time & large # of errors
- Pilot study
  - helps you fix problems with the study
  - do 2, first with colleagues, then with real users

Instructions to Participants

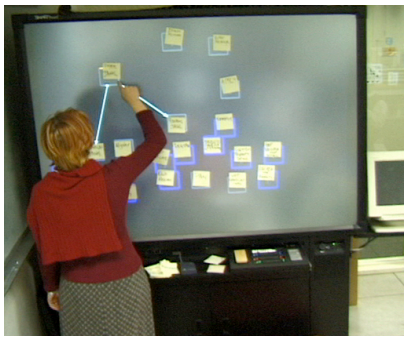
- Describe the purpose of the evaluation
  - "I'm testing the product; I'm not testing you"
- Tell them they can quit at any time
- Demonstrate the equipment
- Explain how to think aloud
- Explain that you will not provide help
- Describe the task
  - give written instructions, one task at a time

Details (cont.)

- Keeping variability down
  - recruit test users with similar background
  - brief users to bring them to common level
  - perform the test the same way every time
    - don't help some more than others (plan in advance)
  - make instructions clear
- Debriefing test users
  - often don't remember, so demonstrate or show video segments
  - ask for comments on specific features
    - show them screen (online or on paper)

Reporting the Results

- Report what you did & what happened
- Images & graphs help people get it!
- Video clips can be quite convincing
  - Especially for convincing other engineers



1 MINUTE BREAK

Discount Usability Engineering

- Reaction to excuses for not doing user testing
  - "too expensive", "takes too long", …
- Cheap
  - no special labs or equipment needed
  - the more careful you are, the better it gets
- Fast
  - on order of 1 day to apply
  - standard usability testing may take a week or more
- Easy to use
  - some techniques can be taught in 2-4 hours

Examples of Discount Usability

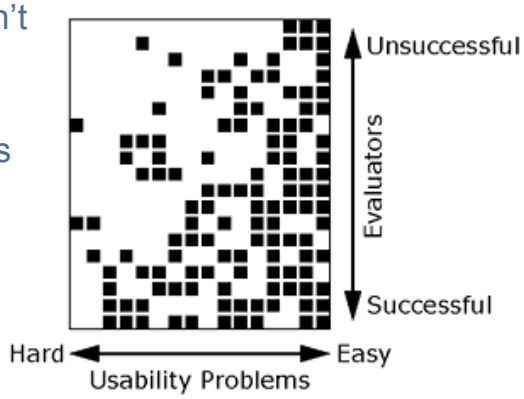
- Walkthroughs
  - put yourself in the shoes of a user
  - like a code walkthrough
- Low-fi prototyping
- Action analysis (GOMS)
- On-line, remote usability tests
- **Heuristic evaluation**

Heuristic Evaluation

- Developed by Jakob Nielsen
- Helps find usability problems in a UI design
- Small set (3-5) of evaluators examine UI
  - independently check for compliance with usability principles (“heuristics”)
  - different evaluators will find different problems
  - evaluators only communicate afterwards
    - findings are then aggregated
- Can perform on working UI or on sketches

Why Multiple Evaluators?

- Every evaluator doesn't find every problem
- Good evaluators find both easy & hard ones



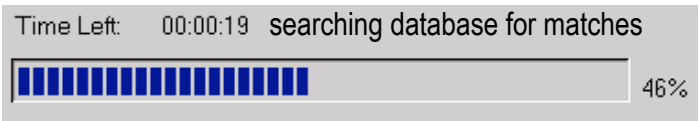
Heuristic Evaluation Process

- Evaluators go through UI several times
  - inspect various dialogue elements
  - compare with list of usability principles
  - consider other principles/results that come to mind
- Usability principles
  - Nielsen's "heuristics"
  - supplementary list of category-specific heuristics
    - competitive analysis & user testing of existing products
- Use violations to redesign/fix problems

Heuristics (original)

H1-1: Simple & natural dialog
H1-2: Speak the users' language
H1-3: Minimize users' memory load
H1-4: Consistency
H1-5: Feedback
H1-6: Clearly marked exits
H1-7: Shortcuts
H1-8: Precise & constructive error messages
H1-9: Prevent errors
H1-10: Help and documentation

Heuristics (revised set)

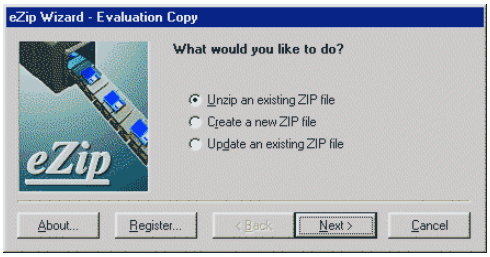


- H2-1: Visibility of system status
  - keep users informed about what is going on
  - example: pay attention to response time
    - 0.1 sec: no special indicators needed, why?
    - 1.0 sec: user tends to lose track of data
    - 10 sec: max. duration if user to stay focused on action
    - for longer delays, use percent-done progress bars

Heuristics (cont.)

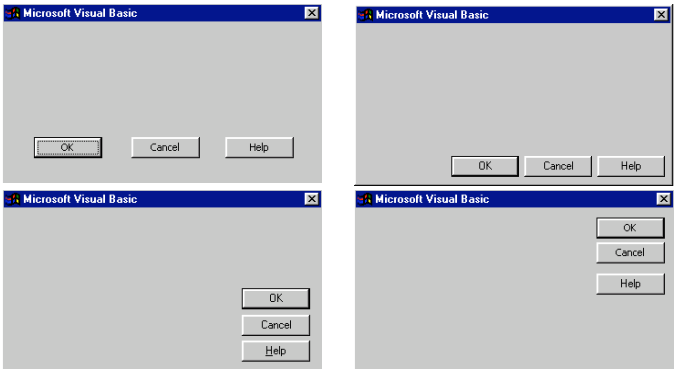
- Bad example: Mac desktop
  - Dragging disk to trash (?)
- H2-2: Match between system & real world
  - speak the users' language
  - follow real world conventions

Heuristics (cont.)

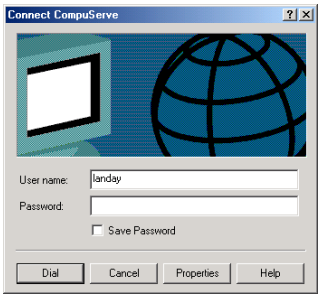


- H2-3: User control & freedom
  - "exits" for mistaken choices, undo, redo
  - don't force down fixed paths
    - like that BART machine…
- Wizards
  - must respond to Q before going to next
  - for infrequent tasks
    - (e.g., modem config.)
  - not for common tasks
  - good for beginners
    - have 2 versions (WinZip)

Heuristics (cont.)



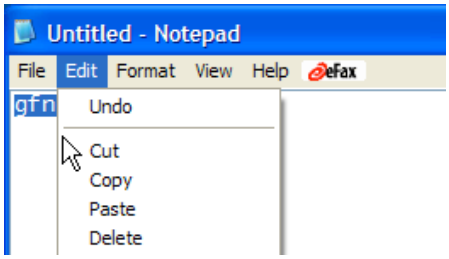
Heuristics (cont.)



- MS Web Pub. Wiz.
- Before dialing
  - asks for id & password
- When connecting
  - asks again for id & pw

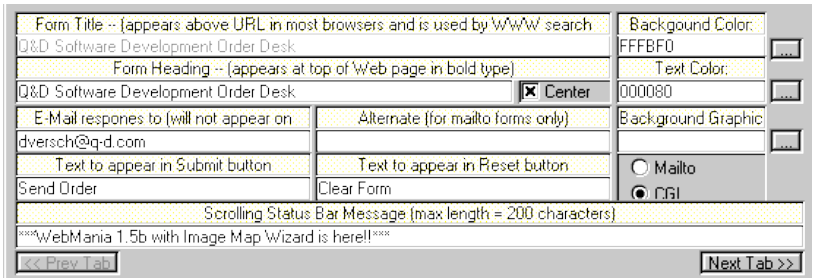
- H2-5: Error prevention
- H2-6: Recognition rather than recall
  - make objects, actions, options, & directions visible or easily retrievable

Heuristics (cont.)



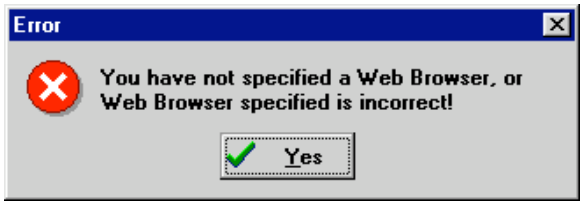
- H2-7: Flexibility and efficiency of use

Heuristics (cont.)



- H2-8: Aesthetic and minimalist design
  - no irrelevant information in dialogues

Heuristics (cont.)



- H2-9: Help users recognize, diagnose, and recover from errors
  - error messages in plain language
  - precisely indicate the problem
  - constructively suggest a solution

Heuristics (cont.)

- H2-10: Help and documentation
  - easy to search
  - focused on the user's task
  - list concrete steps to carry out
  - not too large

Phases of Heuristic Evaluation

1) Pre-evaluation training
   - give evaluators needed domain knowledge and information on the scenario
2) Evaluation
   - individuals evaluate and then aggregate results
3) Severity rating
   - determine how severe each problem is (priority)
     - can do this first individually and then as a group
4) Debriefing
   - discuss the outcome with design team

How to Perform Heuristic Evaluation

- At least two passes for each evaluator
  - first to get feel for flow and scope of system
  - second to focus on specific elements
- If system is walk-up-and-use or evaluators are domain experts, no assistance needed
  - otherwise might supply evaluators with scenarios
- Each evaluator produces list of problems
  - explain why with reference to heuristic or other information
  - be specific and list each problem separately

Examples

- Can't copy info from one window to another
  - violates "Minimize the users' memory load" (H1-3)
  - fix: allow copying
- Typography uses mix of upper/lower case formats and fonts
  - violates "Consistency and standards" (H2-4)
  - slows users down
  - probably wouldn't be found by user testing
  - fix: pick a single format for entire interface

How to Perform Heuristic Evaluation

- Why separate listings for each violation?
  - risk of repeating problematic aspect
  - may not be possible to fix all problems
- Where problems may be found
  - single location in UI
  - two or more locations that need to be compared
  - problem with overall structure of UI
  - something that is missing
    - hard w/ paper prototypes so work extra hard on those
    - note: sometimes features are implied by design docs and just haven't been "implemented" – relax on those

Severity Rating

- Used to allocate resources to fix problems
- Estimates of need for more usability efforts
- Combination of
  - frequency
  - impact
  - persistence (one time or repeating)
- Should be calculated after all evals. are in
- Should be done independently by all judges

Severity Ratings (cont.)

0 - don't agree that this is a usability problem
1 - cosmetic problem
2 - minor usability problem
3 - major usability problem; important to fix
4 - usability catastrophe; imperative to fix

Debriefing

- Conduct with evaluators, observers, and development team members
- Discuss general characteristics of UI
- Suggest potential improvements to address major usability problems
- Dev. team rates how hard things are to fix
- Make it a brainstorming session
  - little criticism until end of session

Severity Ratings Example

1. [H1-4 Consistency] [Severity 3][Fix 0]

The interface used the string "Save" on the first screen for saving the user's file, but used the string "Write file" on the second screen. Users may be confused by this different terminology for the same function.

HE vs. User Testing

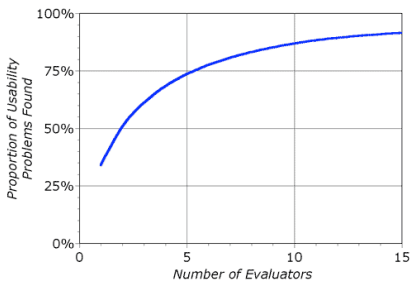
- HE is much faster
  - 1-2 hours each evaluator vs. days-weeks
- HE doesn't require interpreting user's actions
- User testing is far more accurate (by def.)
  - takes into account actual users and tasks
  - HE may miss problems & find "false positives"
- Good to alternate between HE & user testing
  - find different problems
  - don't waste participants

Results of Using HE

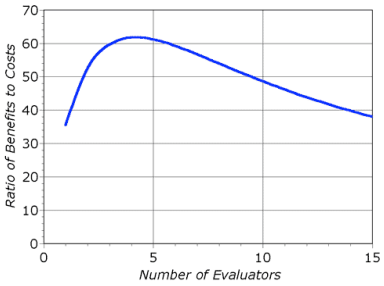
- Discount: benefit-cost ratio of 48 [Nielsen94]
  - cost was $10,500 for benefit of $500,000
  - value of each problem ~15K (Nielsen & Landauer)
  - how might we calculate this value?
    - in-house –> productivity; open market –> sales
- Correlation between severity & finding w/ HE
- Single evaluator achieves poor results
  - only finds 35% of usability problems
  - 5 evaluators find ~ 75% of usability problems
  - why not more evaluators???? 10? 20?
    - adding evaluators costs more & won't find more probs

Decreasing Returns

problems found



benefits / cost



- Caveat: graphs for a specific example

Summary

- User testing
  - important, but takes time (start on mock-ups)
  - use real tasks & representative participants
  - want to know what people are doing & why
    - i.e., collect process data
  - using bottom line data requires more users to get statistically reliable results
- Action-Analysis (GOMS)
  - provides info about important UI properties
  - only gives *performance* for *expert* behavior
  - hard to create model, but easier than user testing
    - changing later is much less work than initial generation

Summary (cont.)

- Heuristic evaluation
  - have evaluators go through the UI twice
  - ask them to see if it complies with heuristics
    - note where it doesn't and say why
  - combine the findings from 3 to 5 evaluators
  - have evaluators independently rate severity
  - alternate with user testing

Further Reading
Evaluation

- Books
  - *Usability Engineering*, by Nielsen, 1994
  - *Handbook of Usability Testing,* by Rubin, 1994
  - *The Design of Sites*, by van Duyne, Landay, & Hong, 2003 (see Appx. A)
  - *The Cartoon Guide to Statistics,* by Gonick and Smith
- Articles
  - "Research Methods in Human-Computer Interaction," by Landauer in *Handbook of Human-Computer Interaction*, M. Helander (ed.), Elsevier, pp. 905-928, 1988.
  - "Discussion of guidelines for user observation", by Kathleen Gomoll and Anne Nichol

Further Reading

Evaluation

- Web Sites & mailing lists
  - useit.com
  - UTEST mail list