

Lecture 18: Usability Engineering Techniques

6.170 Lab in Software Engineering
October 22, 2002

Review: Iterative Design

- Usability engineering is an iterative process
 - Design, implement, evaluate, repeat
- “Spiral model”
 - Use throw-away prototypes and cheap evaluation for first few iterations

Iterative Design: Spiraling Out

- Prototyping tools (increasing fidelity and cost)
 - Paper
 - Storyboards (web pages, Macromedia Director)
 - GUI with stubs (Java, C++, Visual Basic)
- Evaluation tools
 - Heuristic evaluation
 - Formative evaluation
 - Quantitative user studies

Outline

- Usability engineering techniques
 - Iterative design
 - Low-fidelity prototypes
 - Heuristic evaluation
 - Formative evaluation

Low-fidelity Prototypes

- Paper is the fastest prototyping tool
 - Sketch windows, menus, dialogs, widgets
 - Crank out lots of alternatives and evaluate them
- Hand-sketching is OK – even preferable
 - Focus on behavior & interaction, not fonts & colors
- Paper prototypes can even be executed
 - Cut out windows and dialogs
 - Simulate the computer’s responses by moving pieces around and writing on them
 - Photocopier, Post-it notes, transparencies all help

Heuristic Evaluation

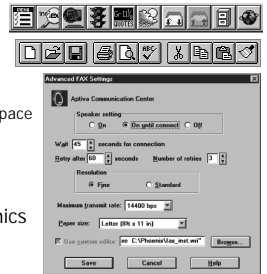
- Start with an “implemented” design
 - Sketch, paper prototype, actual system
- Compare the design against a list of usability heuristics
 - Nielsen’s 10 principles
 - Tognazzini’s 16
 - Mac, Windows, Gnome, KDE guidelines
- List the usability bugs you find

Nielsen's 10 Heuristics

- Less is More
- Speak User's Language
- Minimize Memory Load
- Consistency
- Feedback
- Clearly Marked Exits
- Shortcuts
- Good Error Messages
- Prevent Errors
- Help & Documentation

1. Less Is More

- Minimalist graphic design
 - Use few, well-chosen colors and fonts
 - Remember color blindness
 - Group related items with whitespace
 - Align controls sensibly
- Use concise language
 - Choose labels carefully
- Omit extraneous info & graphics
- Omit unneeded features



2. Speak the User's Language

- Use common words, not techie jargon
 - But use domain-specific jargon where appropriate
- Don't put limits on user-defined names
- Allow aliases/synonyms in command languages
- Metaphors are useful but may mislead



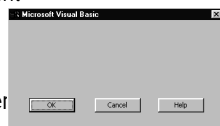
3. Minimize Memory Load

- Allow user to recognize rather than recall (e.g., menus v. commands)
- Use generic commands (e.g., copy & paste)
- All needed information should be visible



4. Consistency

- Principle of Least Surprise
 - Similar things look/act similar
 - Different things look different
- Other properties
 - Size, location, color, wording, ordering, ...
- Command/argument order
 - Prefix vs. postfix
- Follow platform standards



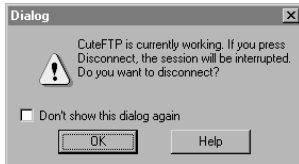
5. Feedback

- Keep user informed of system state
 - Cursor change
 - Selection highlight
 - Status bar
 - Don't overdo it...
- Response time
 - < 0.1 s: seems instantaneous
 - 0.1-1 s: user notices, but no feedback needed
 - 1-5 s: display busy cursor
 - > 1-5 s: display progress bar



6. Clearly Marked Exits

- Provide undo
- Long operations should be cancelable
- All dialogs should have a cancel button



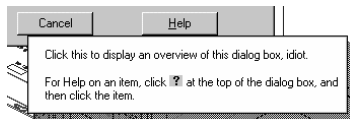
7. Shortcuts

- Provide shortcuts for frequent operations
 - Keyboard accelerators
 - Command abbreviations
 - Styles
 - Bookmarks
 - History



8. Good Error Messages

- Be precise; restate user's input
 - Not "Cannot open file", but "Cannot open file named paper.doc"
- Give constructive help
 - why error occurred and how to fix it
- Be polite and nonblaming
 - Not "fatal error", not "illegal"
- Hide technical details (stack trace) until requested



9. Prevent Errors

- Selection is less error-prone than typing
 - But don't go overboard...



- Disable illegal commands
- Avoid modes
 - Mode = same action has different results
 - Make unavoidable modes spring-loaded

10. Help & Documentation

- Users don't read manuals
 - Prefer to spend time working toward their task goals, not learning about your system
- But manuals and online help are vital
 - Usually when user is frustrated or in crisis
- Help should be:
 - Context-sensitive
 - Task-oriented

Formative Evaluation

- Start with a prototype
- Write up some representative tasks
 - Short, but not trivial
 - e.g., "add this meeting to calendar", "type this letter and print it"
- Find a few representative users
 - 3 is usually enough to reveal obvious problems
- Watch them do tasks with the prototype

How to Watch Users

- Brief the user first (being a test user is stressful)
 - "I'm testing the system, not testing you"
 - "If you have trouble, it's the system's fault"
 - "Feel free to quit at any time"
 - Ethical issues: informed consent
- Ask user to think aloud
- Be quiet!
 - Don't help, don't explain, don't point out mistakes
 - Sit on your hands if it helps
 - Two exceptions: prod user to think aloud ("what are you thinking now?"), and move on to next task when stuck
- Take lots of notes

Watch for Critical Incidents

- Critical incidents: events that strongly affect task performance or satisfaction
- Usually negative
 - Errors
 - Repeated attempts
 - Curses
- Can also be positive
 - "Cool!"
 - "Oh, now I see."

Summary

- You are not the user
- Keep human capabilities in mind
 - Perception, motor, memory, color
- Iterate over your design
- Make cheap, throw-away prototypes
- Evaluate them with heuristics and users

Further Reading

- Low-fidelity prototyping
 - Marc Rettig, "Prototyping for Tiny Fingers", CACM April 1994. <http://doi.acm.org/10.1145/175276.175288>
- Usability heuristics
 - Jakob Nielsen, *Usability Engineering*, AP Professional 1993.
 - Jakob Nielsen, "Heuristic Evaluation." <http://www.useit.com/papers/heuristic/> (Note: Nielsen's web site uses the same heuristics as his book, but with slightly different names.)
 - Bruce Tognazzini, "First Principles." <http://www.asktog.com/basics/firstPrinciples.html>

Futher Reading (2)

- Platform-specific guidelines
 - Java <http://java.sun.com/products/jif/>
 - Gnome <http://developer.gnome.org/projects/gup/hig/>
 - KDE <http://developer.kde.org/documentation/design/ui/>
 - Mac OS <http://developer.apple.com/techpubs/mac/HIGuidelines/HIGuidelines-2.html>
 - MS Windows <http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000443>

Further Reading (3)

- Formative Evaluation
 - Hix & Hartson, *Developing User Interfaces*, Wiley 1995.
- General books on usability
 - Jef Raskin, *The Humane Interface*, Addison-Wesley 2000.
 - Alan Cooper, *About Face*, Wiley 1995.