Heuristic Evaluation

CS160: User Interfaces
John Canny
Topics

• Heuristic Evaluation
• The Process of Heuristic Evaluation
• Pros and Cons of Heuristic Evaluation
Heuristic Evaluation
Usability Heuristics

“Rules of thumb” describing features of usable systems
- Can be used as design principles
- Can be used to evaluate a design

Designed to discover the most common problems

Example: *Minimize users’ memory load*

Pros and cons
- Easy and *inexpensive*
  - Performed by experts
  - No users required
  - Catch many design flaws
- More difficult than it seems
  - Not a simple checklist
  - Cannot assess how well the interface will address user goals
Heuristic Evaluation

Developed by Jakob Nielsen (1994)

Can be performed on working UI or sketches

Small set (3-5) of evaluators (experts) examine UI

– Check compliance with usability heuristics
– Different evaluators will find different problems
– Evaluators only communicate afterwards to aggregate findings
– Use violations to redesign/fix problems
Original Heuristics

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

Also developed by Nielsen.

- Based on factor analysis of 249 usability problems
- A prioritized, independent set of heuristics
Revised Heuristics

H2-1: Visibility of system status
H2-2: Match system and real world
H2-3: User control and freedom
H2-4: Consistency and standards
H2-5: Error prevention
H2-6: Recognition rather than recall
H2-7: Flexibility and efficiency of use
H2-8: Aesthetic and minimalist design
H2-9: Help users recognize, diagnose and recover from errors
H2-10: Help and documentation
Heuristic: Visibility (Feedback)

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
  - 1.0 sec: user tends to lose track of data
  - 10 sec: max. duration if user to stay focused on action
- Short delays: Hourglass
- Long delays: Use percent-done progress bars
  - Overestimate usually better
Heuristic: Visibility (Feedback)

Users should always be aware of what is going on
  – So that they can make informed decision
  – Provide redundant information

Feedback: Toolbar, cursor, ink
Heuristics: Match System & World

H2-2: Match between system & real world

- Speak the users’ language
- Follow real world conventions
- Pay attention to metaphors

Bad example: Mac desktop

- Dragging disk to trash
- Should delete it, **not** eject it
Heuristics: Match System & World

Speak the users’ language (H1-2)

• Withdrawing money at ATM

• Use meaningful mnemonics, icons and abbreviations
H2-3: User control & freedom

- “Exits” for mistaken choices, undo, redo
- Don’t force down fixed paths like BART ticket machine…

Wizards

- Must respond to Q before going to next
- Good
  - For infrequent task (e.g. Internet Config)
  - Beginners (2 versions in WinZip)
- Not good
  - For common tasks
Heuristics: Control & Freedom

• Mark exits: Users don’t like to be trapped!

• Strategies
  – Cancel button (or Esc key) for dialog
    • Make the cancel button responsive!
  – Universal undo
Heuristics: Consistency

H2-4: Consistency and standards
Heuristics: Errors and Memory

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

MS Web Publishing Wizard
  – Before dialing, asks for id & password
  – When connecting, asks again for id & pw
Heuristic: Errors and Memory

• Promote recognition over recall
  – Recognition is easier than recall

• Describe expected input clearly
  – Don’t allow for incorrect input

• Create orthogonal command systems
  – Using generic commands that can be applied to all interface objects
Preventing Errors

Error types

– Mistakes
  • Conscious decision with unforeseen consequences

– Slips
  • Automatic behaviors kicking in
    – Drive to the store, end-up in the office
    – Press enter one time too many…
  • Mode errors
    – Forget the mode the application is in
  • Loss of activation
    – Forget what your goals were
Forcing Functions

Interlock mechanisms
   - Switching from P to D in a car

Lockin mechanisms
   - No eject button for floppy disk on Mac

Lockout mechanisms
   - Exit stairways
Heuristics: Flexibility

H2-7: Flexibility and efficiency of use

- Accelerators for experts (e.g., gestures, shortcuts)
- Allow users to tailor frequent actions (e.g., macros)
H2-8: Aesthetic and minimalist design
   – No irrelevant information in dialogues
Hueristic: Aesthetics

From Cooper’s “The inmates are running the asylum”
Heuristic: Aesthetics

Simple and natural dialog (H1-I)

Present information in natural order

- Occam’s razor
  - Remove or hide irrelevant or rarely needed information
    - They compete with important information on screen
      - Pro: Palm Pilot
      - Against: Dynamic menus
  - Use windows frugally
    - Avoid complex window management

From Cooper’s “About face 2.0”
Heuristics: Help Users

H2-9: Help users recognize, diagnose, and recover from errors

– Error messages in plain language
– Precisely indicate the problem
– Constructively suggest a solution
Good Error Messages

From Cooper’s “About Face 2.0”
Heuristics: Docs

H2-10: Help and documentation

– Easy to search
– Focused on the user’s task
– List concrete steps to carry out
– Not too long
Types of Help

• Tutorial and/or getting started manuals
  – Presents the system conceptual model
    • Basis for successful explorations
  – Provides on-line tours and demos
    • Demonstrates basic features

• Reference manuals
  – Designed with experts in mind

• Reminders
  – Short reference cards, keyboard templates, tooltips…
Types of Help

• Wizards
  – Walks user through typical tasks
    • Users feel they are losing control
    • What if I do not have the information requested?

• Tips
  – Migration path to learning new features
  – Can become boring and tedious
Types of Help

• Context sensitive help
The Process of Heuristic Evaluation
Phases of Heuristic Eval. (1-2)

1) Pre-evaluation training
   - Provide the evaluator with domain knowledge if needed

2) Evaluation
   - Individuals evaluate interface then aggregate results
     • Compare interface elements with heuristics
   - Work in 2 passes
     • First pass: get a feel for flow and scope
     • Second pass: focus on specific elements
   - Each evaluator produces list of problems
     • Explain why with reference to heuristic or other information
     • Be specific and list each problem separately
Phases of Heuristic Eval. (3-4)

3) Severity rating
   – Establishes a ranking between problems
     • Cosmetic, minor, major and catastrophic
   – First rate individually, then as a group

4) Debriefing
   – Discuss outcome with design team
   – Suggest potential solutions
   – Assess how hard things are to fix
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
  – Fix: pick a single format for entire interface

  – Probably wouldn’t be found by user testing
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 evaluations are in

Should be done independently by all judges
Levels of Severity

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
Severity Ratings Example

1. [H2-4 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.
Debriefing

• Conduct with evaluators, observers, and development team members

• Discuss general characteristics of UI

• Suggest improvements to address major usability problems

• Development team rates how hard things are to fix

• Make it a brainstorming session
  – Little criticism until end of session
Pros and Cons of Heuristic Evaluation
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-based testing
- Find different problems
- Don’t waste participants
Why Multiple Evaluators?

• Every evaluator doesn’t find every problem
• Good evaluators find both easy & hard ones
Number of Evaluators

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
    • Many evaluators won’t find many more problems

But always depends on market for product:
  – popular products → high support cost for small bugs
Decreasing Returns

Problems Found

Benefits / Cost

Caveat: Graphs are for a specific example
Benefits 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 $\rightarrow$ productivity
  • Open market $\rightarrow$ sales
  • Customer calls to your customer service center

Tends to find more of the high-severity problems
Summary

• Heuristic evaluation is a discount method
• Have evaluators go through the UI twice
  – Ask them to see if it complies with heuristics
    • Note where it doesn’t and say why
• Have evaluators independently rate severity
• Combine the findings from 3 to 5 evaluators
• Discuss problems with design team

• Cheaper alternative to user testing
  – Finds different problems, so good to alternate
Examples
Revised Heuristics

H2-1: Visibility of system status
H2-2: Match system and real world
H2-3: User control and freedom
H2-4: Consistency and standards
H2-5: Error prevention
H2-6: Recognition rather than recall
H2-7: Flexibility and efficiency of use
H2-8: Aesthetic and minimalist design
H2-9: Help users recognize, diagnose and recover from errors
H2-10: Help and documentation
Powerpoint 2007 issues

- Where did EVERYTHING go???
- File menu
- Undo/redo?
- Format menu (and that thing above it)
- Edit menu
- Master slides (and what is the default design)?
- Standard animations

How do you?

- Insert duplicate slide
- Insert shapes

Is it noun-verb or verb-noun?

- Home vs. Insert menu
Why does this happen?

• The IT industry’s culture is “change is good, status quo is bad,” not enough necessary change in Office.
• But UI’s are secondary languages in which people express their ideas and goals.
• Imagine if we did this with primary languages, such as “English 7.0”:
  – Deprecated: most synonyms and antonyms, “bad”, “very”, “extremely” → “doubleplus ungood”
  – English 7.0 is incompatible with texts in English 4.0 or earlier…
  – Please update your existing texts.
Customizability vs. Usability

This tension goes back to the earliest days of computing.

Engineers and scientists want an “instrument” that behaves exactly as specified.

Most users want to get things done and get on with the rest of their lives.
Usability and Open-Source

Usability is often a problem for OSS projects. Hackers write it, and were the defacto users. Emphasis falls on customizability.

With OSS projects, who pays for usability testing? How do the results propagate to the rest of the community?
Usability and Open-Source

In fact, the companies involved in OSS have pushed usability quite far.
The GNOME desktop was designed for usability first – involved budget evaluation and real user studies.

 GNOME was a leap in usability over other desktops (KDE, XFCE)
Ubuntu Linux

From a Zulu word meaning “humanity”.
Goal is to develop an OS that is accessible and usable by everyone on the planet.
Seems to have worked: Ubuntu is now the most widely used Linux OS.
Based on GNOME with very strong language and accessibility support.
Firefox

From a shakey start, Firefox continues to grab market share.

It has made major improvements in user experience.

Recently hired a team of usability experts from Humanized, including Aza Raskin (son of Apple pioneer Jef).
OLPC (One Laptop per Child)

Use open source software.

Constructionism: Encourage creativity (authoring apps)

Reinvent the UI (Sugar) based on a new “metaphor,” dumps the desktop metaphor.

Novel network code supports mesh networks with and without a server.

Encourage collaboration – apps mesh-aware.
OLPC Reality

No resources devoted to usability or contextual interviews.

Normal Linux apps don’t run, they need to be “sugarized.”

Sugar is interpreted: extremely slow performance.

No help anywhere, for OS or apps, no man pages.

A closed development universe.
OLPC Usability

H2-1: Visibility of system status
H2-2: Match system and real world
H2-3: User control and freedom
H2-4: Consistency and standards
H2-5: Error prevention
H2-6: Recognition rather than recall
H2-7: Flexibility and efficiency of use
H2-8: Aesthetic and minimalist design
H2-9: Help users recognize, diagnose and recover from errors
H2-10: Help and documentation
OLPC Usability - Navigation

The main navigation controls are available from most windows, but auto-hidden with access from corners.
OLPC Usability - Browsing

The web browser activity has very few controls. Most surprisingly, there are no bookmarks per se.
You can “bookmark” by clicking the right star, which adds the link to the Journal.
OLPC Usability - Browsing

The journal is like an event log. You can put things there, and app start/stop notices go there.
You find things in the journal by time, or from the tags that have been added to items there.

<table>
<thead>
<tr>
<th>OLPC Library</th>
<th>13 minutes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OLPC Library</td>
<td>17 minutes</td>
<td></td>
</tr>
<tr>
<td>OLPC Library</td>
<td>20 hours and 55 minutes</td>
<td></td>
</tr>
<tr>
<td>Terminal Activity</td>
<td>20 hours and 55 minutes</td>
<td></td>
</tr>
<tr>
<td>Terminal Activity</td>
<td>21 hours and 14 minutes</td>
<td></td>
</tr>
<tr>
<td>OLPC Library</td>
<td>9 minutes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>21 hours and 42 minutes</td>
<td></td>
</tr>
<tr>
<td>Terminal Activity</td>
<td>2 months and 2 weeks</td>
<td></td>
</tr>
<tr>
<td>Terminal Activity</td>
<td>2 months and 2 weeks</td>
<td></td>
</tr>
<tr>
<td>Amazon.com: eee pc</td>
<td>5 minutes</td>
<td></td>
</tr>
<tr>
<td>All About My Body!</td>
<td>6 months and 4 days</td>
<td></td>
</tr>
</tbody>
</table>
Once you find an item, you can start a new web browser with that link in it.
OLPC Sound

• There is no sound mixer console, but the sound chip has the usual internal mixer parameters.
• A little tinkering will often turn sound off.
• A little more tinkering will turn sound off – for good.

• The fix? Go to the terminal window and type:

  amixer set PCM 100 on
How did this happen?
Nicolas Negroponte (OLPC founder):

"In fact, one of the saddest but most common conditions in elementary school computer labs (when they exist in the developing world), is the children are being trained to use Word, Excel and PowerPoint," Negroponte wrote in an e-mail interview. "I consider that criminal, because children should be making things, communicating, exploring, sharing, not running office automation tools."
OLPC difficulties

Things heading down the wrong track when designers declare what users should be doing, instead of asking and observing them.

Things are likely to go wrong when purchasers (governments and school boards) are different from users (children, teachers and parents).
OLPCs Shipping

OLPC Peru Activity Pack:

- org.laptop.AbWordActivity (MS Word replacement)
- org.codewiz.XaoS
- org.laptop.WebActivity
- org.laptop.Oficina…

Nepal Laptops
- Ship with Adobe Flash Player
OLPC Adds Windows XP To XO Laptop

Microsoft will charge OLPC a highly reduced Windows XP licensing fee of $3 per unit under a program it calls Unlimited Potential.

By Paul McDougal
InformationWeek
May 16, 2008 01:50 PM

Microsoft (Nasdaq: MSFT) said Thursday that it has reached a deal with the One Laptop Per Child project to make its Windows XP operating system available on the group's low-cost notebook computers for distribution to students in developing countries.

"By supporting a wide variety of affordable computing solutions for education that includes OLPC's XO laptop, we aim to make technology more relevant, accessible, and affordable for students everywhere," Craig Mundie, Microsoft's chief research and strategy officer, said in a statement.
Summary

Usability heuristics make evaluation much more cost-effective.

How to conduct a heuristic evaluation.
Best to use in conjunction with real user studies.

Industry and OSS forces often push against usability.
But those efforts that push back have been consistently successful.