Discount Usability, Heuristic Evaluation

COMM/INFO 3450

Heavy borrowing from Loren Terveen's CS 5115
Heuristic evaluation

- Idea: many problems are universal; use a manageable set of rules of thumb

- Idea: vs. cognitive walkthrough, remove focus on tasks: consider whole interface

- Idea: combine individual inspections
What an evaluator does

- Each person looks at the UI at least twice
  - Once to get an overall feel for the system
  - Second to inspect interface elements and consider them in terms of the heuristics

- May go by screen, by heuristic, or by overall feel
  - Recommended for first-timers: by heuristic
Ten Heuristics

- Simple and natural dialog
- Speak the user's language
- Reduce user memory load
- Be consistent
- Provide feedback
- User control and freedom
- Flexibility and efficiency of use
- Help users recognize, diagnose, and recover from errors
- Error prevention
- Help and documentation
1. Simple and natural dialog

- Exploit the user’s conceptual model
- Match tasks naturally
  - A place for good metaphors
Simple and natural dialog

- Info should appear in natural order
  - E.g., data entry for forms

- Hide irrelevant or rarely needed info
  - Less is more
  - Easier to learn, fewer errors, less distraction

- Good graphic design
2. Speak the user's language

- Use terminology based on user’s domain
- Avoid engineering jargon
- Use the user’s native language
- Use conventional meanings
- View the interaction from the user’s perspective
Search Results

Nothing Found

Your search for [interface user evaluation heuristic] did not return any results.

You may also revise it and try your search again below or click advanced search for more options.

interface user evaluation heuristic

[Advanced Search] [Search Help/Tips]

The following characters have specialized meaning:

<table>
<thead>
<tr>
<th>Special Characters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>, ( ) [</td>
<td>These characters end a text token.</td>
</tr>
<tr>
<td>= &gt; &lt; !</td>
<td>These characters end a text token because they signify the start of a field operator. ( ! is special: ! = ends a token. )</td>
</tr>
<tr>
<td>@ \ Q &lt; [</td>
<td>These characters signify the start of a delimited token. These are terminated by the end character associated with the start character.</td>
</tr>
</tbody>
</table>
1. How to guarantee a girl never leaves you. [comic] (aa1.naurunappula.com) 311 points posted 10 hours ago by bennmu 172 comments

2. BBC: "CIA role claim in Kennedy killing. New video and photographic evidence that puts three senior CIA operatives at the scene of Robert Kennedy's assassination has been brought to light." [politics] (news.bbc.co.uk) 194 points posted 11 hours ago by georgewashingtonblog 91 comments

3. USA 1939 - 1969 [Slow Loading Pics] [pics] (oreys.com) 64 points posted 7 hours ago by nechronin 92 comments

4. the missed opportunity of acid 3 [programming] (shaver.off.net) 24 points posted 3 hours ago by davidhassen10 of 3 comments

5. The Dow Is Crashing - when measured in sound money [business] (financialsense.com) 50 points posted 7 hours ago by SLAUTCANS 61 comments

6. India Calls Out the World on Food-Based Bio-Fuels [science] (news.bbc.co.uk) 61 points posted 8 hours ago by Eulpian 112 comments

7. NBC: Experts Say Sirhan Sirhan Did Not Shoot RFK [worldnews] (youtube.com) 22 points posted 3 hours ago by ireaditdido 1 comment

8. Video of Hillary Clinton dodging bullets, bombs and busting out a sidearm revealed! [funny] (electiongeek.com)
3. Reduce User Memory Load

- Recognition over recall

- Menus, icons, constrained input
  - Versus command lines and free text fields
  - Best of both worlds: natural command lines

- Use examples

- Needed information should be visible
How to make better?

Font:
- Symbol

Font style:
- Italic

Size:
- 32

Effects:
- Underline
- Superscript
- Shadow
- Offset: 0
- Emboss
- Subscript

Color:
- Default

This is a TrueType font. This same font will be used on both your printer and your screen.

Fonts displayed in their fonts:
F2, F3, or F4?
Previews FTW
4. Be consistent

- Consistent with:
  - Task
  - User model
  - User experience (e.g., other interfaces)
- Consistent within an application
- Consistent across applications
  - The Apple Look and Feel
What should be consistent?

- Location of information
- Language / graphics
- Terminology
- Layout
- Input syntax
- Command effects and behavior
- Shortcuts
- Organization of commands
5. Provide Feedback

- System should continuously inform users what it is doing, how it interprets actions

- Response Time
  - < 0.1 sec – seems instantaneous
  - 0.1-10 sec – noticeable, but doesn’t disrupt train of thought
  - > 10 sec – users want to work on other tasks
Good Feedback

![Software Interface for Download Accelerator Plus](image)

- **Saving To:** D:\download\WebFerret.exe
- **Status:** Receiving file (4)
- **Resume:** Resume Supported
- **URL File:** ftp://ftp.download.com/pub/zed/WebFerret.exe
- **File Info:**
  - **Transfer Rate:** 131.1 KB/Sec
  - **Time Left:** 2 Second(s)
  - **File Size:** 291.6 KB of Unknown

**Simultaneous Connections**:

<table>
<thead>
<tr>
<th>#</th>
<th>Size</th>
<th>Connection Status</th>
<th>Server Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>66.8 KB</td>
<td>Receiving file</td>
<td>ftp.download.com</td>
</tr>
<tr>
<td>2</td>
<td>69.6 KB</td>
<td>Receiving file</td>
<td>ftp.download.com</td>
</tr>
<tr>
<td>3</td>
<td>119.0 KB</td>
<td>Receiving file</td>
<td>ftp.download.com</td>
</tr>
<tr>
<td>4</td>
<td>36.2 KB</td>
<td>Receiving file</td>
<td>ftp.download.com</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>
Good (subtle) feedback

What did I select?

What mode am I in?

How is the system interpreting my actions?

Too subtle?
Let's evaluate

- Let's do Campus to Campus reservations
- 15 minutes, groups of 3-5, 1-2 heuristics each
- Which element(s)? Which heuristic(s)?
  - No fixes. Not yet.

Simple and natural dialog
Speak the user's language
Reduce user memory load
Be consistent
Provide feedback
6. User control and freedom

- Users do not like to feel trapped
- Support exploration
- Restore defaults
- Universal undo
- Cancellability
Escaping from a wizard: good
7. Flexibility & efficiency of use

7. Provide shortcuts

Experience users should be able to do frequent, familiar operations fast.

Keyboard and mouse accelerators
- Function keys
- Menu shortcuts
- Command completion
- Double-clicking to invoke default action

- Type-ahead
- Macros and scripting
- Reusable command/edit history
  - ~60% of page visits are revisits

Recently used files

Right-click brings up menu

Keyboard accelerators for menus

Customizable toolbars and palettes for frequent actions

Scrolling within a page or by whole pages
No keyboard shortcuts? Why?!?
Flexibility & efficiency of use

- Support experienced users, too
  - Keyboard and mouse accelerators
  - Command/filename completion

- Type-ahead

- Macros and scripting
Flexibility & efficiency of use

- Reusable command/edit history
  - ~60% of page visits are revisits
  - Personalizing the interface often good
8. Help users... errors

- Use clear, simple, and polite language
- Be specific about the problem
- Offer possible solutions
- Turn it into a learning experience
  - Point to help
  - Encourage exploration
This page is accessing information that is not under its control. This poses a security risk. Do you want to continue?

Internet Explorer

This page is accessing information that is not under its control, which poses a security risk. For example, this page might have unauthorized access to any sensitive information you enter.

Learn more about potential security risks affecting this page.

You should not continue if you do not know or trust the developer of this page.

Do you want to continue?
Bad error messages

Sorry, a system error occurred.
unimplemented trap
To temporarily turn off extensions, restart and hold down the shift key.

java.lang.NumberFormatException: null
java.lang.Integer.parseInt(Integer.java:415)
java.lang.Integer.parseInt(Integer.java:497)
guestDBServlet.doGet(guestDBServlet.java:21)
javax.servlet.http.HttpServlet.service(HttpServlet.java:689)
javax.servlet.http.HttpServlet.service(HttpServlet.java:802)
9. Prevent Errors

- Remember error types?
- Make things distinct
- Don't give misleading signals
  - Appropriate controls
  - Greying out when appropriate
Feedback to prevent an error

Save As...

Y:\Teaching\5115\Fall 2003\lectures\15-2003-10-21.ppt already exists.
Do you want to replace it?

Yes  No
(Better) feedback

This folder already contains a file named 'group-2003-submitted.doc'.

Would you like to replace the existing file

187 KB
modified: Today, October 21, 2003, 1:13:37 PM

with this one?

187 KB
modified: Friday, May 30, 2003, 9:03:15 AM

Yes  No
10. Help and Documentation

- Most users do not read the manual
  - When users are reading the manual, they probably are in a panic!
  - Make documentation task oriented

- Document in the context of use
  - Tooltips and context menus
  - Others’ activity (read and edit wear)
# Labels, examples, constraints

<table>
<thead>
<tr>
<th>Genres</th>
<th>Action</th>
<th>Adventure</th>
<th>Animation</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Comedy</td>
<td>Crime</td>
<td>Documentary</td>
<td>Drama</td>
</tr>
<tr>
<td></td>
<td>Fantasy</td>
<td>Film-Noir</td>
<td>Horror</td>
<td>IMAX</td>
</tr>
<tr>
<td></td>
<td>Musical</td>
<td>Mystery</td>
<td>Romance</td>
<td>Sci-Fi</td>
</tr>
<tr>
<td></td>
<td>Thriller</td>
<td>War</td>
<td>Western</td>
<td></td>
</tr>
</tbody>
</table>

Was: Drama, Thriller

<table>
<thead>
<tr>
<th>Language</th>
<th>English</th>
</tr>
</thead>
</table>

Was: same

<table>
<thead>
<tr>
<th>Director</th>
<th>Mick Jackson</th>
</tr>
</thead>
</table>

Was: same

<table>
<thead>
<tr>
<th>Starring</th>
<th>Stephen Dorff, Mira Sorvino, Anjelica Huston</th>
</tr>
</thead>
</table>

Was: same

<table>
<thead>
<tr>
<th>Release dates (U.S.)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Theatre: 2006-04-09</td>
<td>Was: 0000-00-00</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DVD: 2006-07-25</td>
<td>Was: same</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>VHS: 0000-00-00</td>
<td>Was: same</td>
</tr>
</tbody>
</table>

**How to format:** The best format is "YYYY-MM-DD" for dates, although we try to understand others.

*OK:* Jan 11, 2000 or 01/11/2000

*Best:* 2000-01-11
Those heuristics again

- Simple and natural dialog
- Speak the user's language
- Reduce user memory load
- Be consistent
- Provide feedback
- User control and freedom
- Flexibility and efficiency of use
- Help users recognize, diagnose, and recover from errors
- Error prevention
- Help and documentation
Doing the evaluation

- 3-5 evaluators do independent evaluations
  - Different evaluators find different problems

- Evaluators then merge their findings

- Collectively rate severity of the problems

- Debriefing/brainstorming
  - How to fix the problems
  (and point out what’s really good)
Why multiple evaluators?

Diagram showing a scatter plot with axes for Evaluators and Usability Problems. The plot is divided into Successful and Unsuccessful categories.

- Evaluators
- Usability Problems
- Successful
- Unsuccessful
- Hard
- Easy
Heuristic Evaluation Outputs

- **Individuals:**
  - List of problems
  - For each problem, how found/what heuristics were violated

- **Group:**
  - Aggregated list of problems and provenance
  - Severity ratings
  - Annotations, maybe including possible solutions
Severity ratings

- Used to allocate resources to fix problems
- Based on
  - **Frequency** the problem will occur
  - **Impact** of problem (hard or easy to overcome)
  - **Persistence** (will users learn a work around or will they be bothered every time?)

1 – cosmetic problem
2 – minor usability problem
3 – major usability problem; important to fix
4 – usability catastrophe – must fix
# One representation

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
<th>Reasoning (+heuristic)</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>On signin</td>
<td>Splash screen requests focus no matter what the user is doing.</td>
<td>User control and freedom. We don’t want to waste people’s time whenever they start our product.</td>
<td>3</td>
</tr>
<tr>
<td>Tray icon</td>
<td>Icon flashes once when you receive a message.</td>
<td>Visibility of system status. The flash is too easy to miss.</td>
<td>2</td>
</tr>
<tr>
<td>Buddy screen</td>
<td>No way to sort contacts by frequency of use</td>
<td>Flexibility and efficiency of use. Users will want easy access to their best buds.</td>
<td>2</td>
</tr>
</tbody>
</table>
Not the only heuristics

- Different people, different wisdom

- May vary by domain or platform
  - Voice interface guidelines, or cell phones, might be considerably different than large-screen WIMP
  - “Interesting exercise”: how might the heuristics be different for groupware? On small screens?
A question to ask

- How do we know this is a good technique?
- What is a good technique?
Evaluation dimensions

- Formality
- Generality
- Cost
- Usefulness in design (!)
- Effectiveness
Effective?

- Find errors quickly
- Find many errors
- Find serious errors
- Find commonly encountered errors
- Find diverse errors
Heuristic eval

- Formality: low to middling
- Generality: high
- Cost: fairly low
- Effectiveness: Many errors quickly, but maybe not the most serious/common errors; multiple evaluators help identify span of errors
Cognitive walkthroughs

- Formality: varied, but fairly high
- Generality: technique general; specific instances very not general
- Cost: fairly high
- Effectiveness: Finds some errors, including some likely to be encountered, but high-level blind and high cost per error
Heuristic walkthrough

- “Best of both worlds”

- Idea: combine the great taste of heuristic freedom with the guidance of focusing on common user tasks

- Two passes: task-focused, then gestalt
Cognitive jogthrough

- Idea: keep tasks, reduce structure
- Much less formal
- More general: generate actions on the fly
- Lower-cost
- Still reasonably effective, plus can explore more paths than walkthroughs
Main takeaways

- Discount usability methods can improve usability relatively cheaply
- Heuristic evaluation very useful
- With HE, or any other method, think through the value it adds to your project.
  - It’s okay to prefer some methods
  - Combining methods provides advantages
  - Using methods serially also good
Bonus slides follow

- Leftovers and castaways from Loren’s deck
Benefits of a Cognitive Walkthrough

- Focus most on first experiences - learnability
- Easy to learn
- Can do early in the software cycle
- Surfaces and examines assumptions about what users might be thinking.
- Can identify controls that are obvious to the designer but not to the user
- It can suggest difficulties with labels and prompts
- It can help find inadequate feedback
- Can help find inadequacies in the spec
When to do a Cognitive Walkthrough

- Before formal evaluation with your users
  - Don’t waste their time!
- On your own, for small pieces of the whole
- Can do a walkthrough of a complete task as the interface develops
Cognitive Walkthrough How To -

- Prototype interface (preferably LoFi)
- Task description
- Scenario – written list of the actions to complete the task in the interface
- An idea of who the users will be and their characteristic (so you can tell believable stories)
  - Personas may be useful
For each action in the sequence
  ▪ tell the story of why the user will do it
  ▪ ask critical questions (recall 7 Stages of Action)
    ▪ Will users be trying to produce the effect? I.e., will they form the goal designers wanted them to?
    ▪ Will users see the correct control?
    ▪ Will users recognize that this is the control they’re after, i.e., that it will advance them toward their goal?
      ▪ Or will they select a different control instead?
    ▪ Will users understand the feedback? That is, will be they be able to tell that they achieved their intended goal or at least made progress toward it?
A Quick Example: PGP

- Users do not understand the key metaphor
- Users do not understand the implications of signing
- Signature verification is not represented visually

From “Why Johnny Can’t Encrypt: A Usability Evaluation of PGP 5.0”, Alma Whitten & J.D. Tygar
Graphic Design Principles

- Group related items
- Make important items stand out
- Color
  - Don’t over-do it
  - Make sure the interface works without it
  - Use color to categorize, differentiate, highlight – not to give information
Grouping
A theory of (new) users

- Based on cognitive psychology and 7 stages...

- The user sets a goal to be accomplished with the system (for example, "check spelling of this document").

- The user searches the interface for currently available actions (menu items, buttons, command-line inputs, etc.).
A theory of (new) users

- The user selects the action that seems likely to make progress toward the goal.
- The user performs the selected action and evaluates the system's feedback for evidence that progress is being made toward the goal.

- Note, nothing about trying new things, exploring, reading help, etc.
Empirical Support

1. Users try label-guided actions first before they try direct manipulations of unlabeled objects.
2. Users are reluctant to try atypical actions.
3. Users are reluctant to extend their search beyond the readily available menus and controls.
Exploiting the model, cheaply

- Cognitive walkthroughs
  - Explicitly follow the model
  - Pretend to be new users
  - Use scenarios to uncover flaws in interaction

- Heuristic evaluation
  - Implicitly encode the model
  - Derived from common usability flaws
  - Claimed to be cheap, easy to learn
Cognitive Walkthroughs

- What if we run with the theory?
- A task-oriented method of evaluating an interface *without* users
- A formalized way of imagining people’s thoughts and actions when they *(first)* use an interface.
Cognitive Walkthroughs

- Goals
  - Evaluate choice-points in the interface
  - Find confusing labels, icons, images, options
  - Detect likely user navigation errors
- Remember: improvement, not defense
- Start with a complete TCUID scenario
  - Low-level, action-by-action scenarios
Cognitive walkthrough roles

- **The Presenter:** *walk through the scenario*
  - What actions does the user take?
  - *Why* do they take those particular actions?
- **The Questioners:** *tear it apart*
  - Why that button? Will they see the relevant info?
  - More generally, use the seven stages of action
- **The Scribe:** *record found problems*
- Rotate the roles, and *don’t go solo*
The mindset

- Tell a believable story
  - How to accomplish the task, action-by-action?
  - Based on user knowledge and system interface
  - Recall DOET principles (Is this visible? Is feedback clear? Is there a gulf of execution? ...)

- Be highly skeptical
  - remember the goal: finding problems

- Every gap is an interface problem
Example: Going Backwards

- I’ll be The Presenter (ha!)
- Scenario: going backwards one slide during a presentation.
Ex: Does 345 have a prerequisite?

- Exercise: Split into groups of 4.
- Then into groups of 2, “COMM” and “INFO”.
  - Your group of 2 will be the presenter and the scribe for the task of finding whether 345 has a prereq using your department’s website.
  - Spend 3 minutes constructing your scenario.
- Present to the other group, who are your Questioners, for 3 minutes. Scribe, take notes.
- Switch groups.
Shortcomings of CogWalk

- Is diagnostic, not prescriptive
- Focuses mostly on novice users
- Relies on the ability of designers to put themselves in the user’s shoes (tsk!)
- Costly to develop click-level scenarios
  - Not all users have the same model
  - Many tasks have multiple reasonable paths
Why iterate and evaluate?

- Explore options
  - Trying multiple options is good practice
  - Some questions are empirical

- Manage risk
  - Learn more, faster
  - Catch errors, sooner
Why, continued

- React quickly
  - Why find the same error three times?
  - Allows greater user involvement, maybe
  - With some techniques, always a releasable design/product

- Increase chances of success
Why not?

- Management issues
- Habit issues
- Pride issues
- Cost issues
Cost does matter

- User testing can be expensive and fussy
- Many companies/people question value
  - This is part of the “why HCI” problem
  - Nielsen has some cost-benefit examples

- Goal: develop cheap techniques to improve usability, preferably without users
Discount usability

- Evaluation on the cheap
  - Cognitive walkthroughs – lower user costs
  - Paper prototypes – don’t develop (yet)

- Evaluation based on expertise
  - Heuristics – collected, distilled wisdom
  - Guidelines/standards – situational rules