New Assignment

Perform Heuristic Evaluation of another student’s Programming Assignment #4

Due: 1 week from today
10 points, no extra credit

Section this week

1. Bring your paper prototype if you need practice being the “computer”
2. Work on the heuristic evaluation assignment
Midterm on 3/17

If you are registered with the DSP office and have special needs, we need to see your letter by this Wednesday, 3/10, 6pm to make accommodations!

Today

1. Keystroke Level Model (KLM) Example
2. Qualitative Evaluation: Cognitive Walkthrough and Heuristic Evaluation

Keystroke Level Model (KLM)

Describe the task using the following operators:

K: pressing a key or a pressing (or releasing) a button
\( t_K = 0.08 - 1.2s \) (0.2 good rule of thumb)

P: pointing
\( t_P = 1.1s \) (without button press)

H: Homing (switching device)
\( t_H = 0.4s \)

D(n,l): Drawing segmented lines
\( t_D = 0.9n + .16l \)

M: Mentally prepare
\( t_M = 1.35s \)

R(t): system response time
\( t_R = t \)

KLM Heuristic Rules (Raskin’s)

0: Insert M in front of all K
In front of all P’s selecting a command (not in front of P’s ending command)

1: Remove M between fully anticipated operators
PMK → PK

2: if a string of MKs belong to cognitive unit delete all M but first
4664.23: MKMKMKMKMKMKMKMK → MKKKKKKK

3: if K is a redundant terminator then delete M in front of it
del MKMK → MKK

4a: if K terminates a constant length string (command name) delete the M in front of it
cd MKMK → MKK

4b: if K terminates a variable length string (parameter) keep the M in front of it
cd class MKKKMKKKKKMK → MKKKMKKKKKMK
Using KLM

Encode using all physical operators (K, P, H, D(n,l), R(t))

Apply Raskin’s KLM rules [0-4]

Transform R followed by an M

If $t < t_M: R(t) \rightarrow R(0)$

If $t_M < t: R(t) \rightarrow R(t - t_M)$

Compute the total time by adding all individual times

Converting Temperatures I

**Temperature World**
Temperature Converter - Plus!™

Type the temperature value to be converted in the “F”, “C”, or “K” box and click the submit button.

Fahrenheit | Celsius | Kelvin | Submit | Clear

Assume: Focus is on the Fahrenheit box, so typing on the keyboard will enter text directly into that box.

Case 1 (F->C): MKKKK HMPK = 5.2s

Case 2 (C->F): H MPK H MKKKK HMPK = 8.65s

Average: 6.925s

Converting Temperatures 2

**Google**

92.3°F in C

Google Search | I'm Feeling Lucky

92.3 degrees Fahrenheit = 33.5 degrees Celsius

More about calculator.

Assume: Focus is on the search box, so typing on the keyboard will enter text directly into that box.
Converting Temperatures 2

Google

62.3°F in C

Is TemperatureWorld always preferable?

We looked at one isolated task – do you need to “reset” UI for next conversion? What about interleaving with other tasks?

We assumed desktop input devices (Mouse + Keyboard). What about mobile input?

What about errors?

What GOMS Can Model

Task must be goal-directed

Some activities are more goal-directed

Creative activities may not be as goal-directed

Task must be a routine cognitive skill

As opposed to problem solving

Good for things like machine operators

Serial & parallel tasks (CPM-GOMS)

Advantages of GOMS

Gives qualitative & quantitative measures

Model explains the results

Less work than user study – no users!

Easy to modify when UI is revised

Research: Need tools to aid modeling process since it can still be tedious
Disadvantages of GOMS

- Not as easy as other evaluation methods
- Takes lots of time, skill, & effort
- Only works for goal-directed tasks
- Assumes tasks **expert** performance without **error**
- Does not address other important UI issues, e.g., readability, memorizability of icons, commands

Usability Inspection Methods

Iterative Design

- **Design**
  - Brainstorming
  - Task analysis
  - Contextual inquiry
- **Prototype**
  - Low-fi, paper
- **Evaluate**
  - Low-fi testing,
  - **Qualitative eval**
  - Quantitative eval

Genres of assessment

- **Automated** Usability measures computed by software
- **Empirical** Usability assessed by testing with real users
- **Formal** Models and formulas to calculate measures
- **Inspection** Based on heuristics, skills, and experience of evaluators
Quantitative Testing is Costly

User studies are very expensive – you need to schedule (and normally pay) many subjects.

User studies may take many hours of the evaluation team’s time.

A user test can easily cost $10k’s

“Discount Usability” Techniques

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)

Easy to use
Can be taught in 2-4 hours

“Discount Usability” Techniques

Cognitive Walkthroughs
Put yourself in the shoes of a user
Like a code walkthrough

Heuristic Evaluation
Assess interface based on a predetermined list of criteria

Other, non-inspection techniques are on the rise
e.g. online remote experiments with Mechanical Turk

Cognitive Walkthrough
Cognitive Walkthrough

Formalized technique for imagining user’s thoughts and actions when using an interface:

“Cognitive walkthroughs involve simulating a user’s problem-solving process at each step in the human-computer dialog, checking to see if the user’s goals and memory for actions can be assumed to lead to the next correct action.” (Nielsen, 1992)

Cognitive Walkthrough Example

**Task:** Find the call number and location of the latest edition of the book “Interaction Design” by Preece, Rogers & Sharp in the Berkeley library

**Typical users:** Students who are familiar with the web, but not necessarily with the library or its website

Cognitive Walkthrough Example

Given an interface prototype or specification, need:

- A detailed task with a concrete goal, ideally motivated by a scenario
- Action sequences for user to complete the task

Ask the following questions for each step:

- Will the users know what to do?
- Will the user notice that the correct action is available?
- Will the user interpret the application feedback correctly?

Record: what would cause problems, and why?
Cognitive Walkthrough Example

Step 2: Complete the search form

Will the user know what to do?

Will user notice that action is available?

Will user interpret feedback correctly?

Cognitive Walkthrough Example

Step 3: Locate the right edition, click to detail screen

Will the user know what to do?

Will user notice that action is available?

Will user interpret feedback correctly?

Cognitive Walkthrough Example

Step 4: Locate call number and library location

Will the user know what to do?

Will user notice that action is available?

Will user interpret feedback correctly?

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

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 on sketches

Small set (3-5) of evaluators (experts) examine UI
Evaluators check compliance with usability heuristics
Different evaluators will find different problems
Evaluators only communicate afterwards to aggregate findings
Designers use violations to redesign/fix problems

Nielsen’s Ten 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, recover from errors
H2-10: Help and documentation

Original Heuristics

H1-1: Simple and 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
H2-1: Visibility of system status

Keep users informed about what is going on. Example: 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

H2-2: Match System & World

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

Bad example: Mac desktop
H2-2: Match System & World

H2-3: User control & freedom

Users don’t like to be trapped!

Strategies
- Cancel button (or Esc key) for dialog
- Make the cancel button responsive!
- Universal undo

Offer “Exits” for mistaken choices, undo, redo
Don’t force the user down fixed paths

Wizards
- Must respond to Q before going to next step
- Good for infrequent tasks (e.g., network setup) & beginners
- Not good for common tasks (zip/unzip)
H2-4: Consistency and standards

H2-4: Consistency and Standards

H2-5: Error Prevention

H2-5: Error Prevention
H2-5: Error Prevention

Don't allow incorrect input

Preventing Errors

Error types

Slips
User commits error during the execution of a correct plan.

Types:
- Habitually answer "no" to a dialog box
- Forget the mode the application is in

Mistakes
User correctly executes flawed mental plan

Usually the result of a flawed mental model – harder to guard against

H2-6: Recognition over Recall

Minimize the user's memory load by making objects, actions, and options visible.
H2-7: Flexibility and efficiency of use

H2-8: Aesthetic and minimalist design

H2-8: Aesthetic and minimalist design

No irrelevant information in dialogues

H2-8: Aesthetic and minimalist design

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 Pilots
Against: Dynamic menus
Use windows frugally
Avoid complex window management
H2-8: Aesthetic and minimalist design

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

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

H2-10: Help and documentation

Help should be:
- 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

Context sensitive help

Search
### 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**

- **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 and Persistence
- Should be calculated after all evaluations are in
- Should be done independently by all judges

### Levels of Severity

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>don't agree that this is a usability problem</td>
</tr>
<tr>
<td>1</td>
<td>cosmetic problem</td>
</tr>
<tr>
<td>2</td>
<td>minor usability problem</td>
</tr>
<tr>
<td>3</td>
<td>major usability problem; important to fix</td>
</tr>
<tr>
<td>4</td>
<td>usability catastrophe; imperative to fix</td>
</tr>
</tbody>
</table>

### Severity Ratings Example

1. [H2-4 Consistency] [Severity 3]

   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**
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

Why not more evaluators???? 10? 20?
Adding evaluators costs more
Many evaluators won’t find many more problems

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 one specific example!

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

Next Time

Quantitative Evaluation