Human Computer Interaction

Designing Systems that work for People
design rules

Designing for maximum usability
– the goal of interaction design

- Principles of usability
  - general understanding

- Standards and guidelines
  - direction for design
types of design rules

- principles
  - abstract design rules
  - low authority
  - high generality

- standards
  - specific design rules
  - high authority
  - limited application

- guidelines
  - lower authority
  - more general application
Principles to support usability

Learnability
the ease with which new users can begin effective interaction and achieve maximal performance

Flexibility
the multiplicity of ways the user and system exchange information

Robustness
the level of support provided the user in determining successful achievement and assessment of goal-directed behaviour
Principles of learnability (1)

Predictability
- determining effect of future actions based on past interaction history
- operation visibility

Synthesizability
- assessing the effect of past actions
- immediate vs. eventual honesty
Principles of learnability (2)

Familiarity
- how prior knowledge applies to new system
- guessability; affordance

Generalizability
- extending specific interaction knowledge to new situations

Consistency
- likeness in input/output behaviour arising from similar situations or task objectives
Principles of flexibility (1)

Dialogue initiative
- freedom from system imposed constraints on input dialogue
- system vs. user pre-emptiveness

Multithreading
- ability of system to support user interaction for more than one task at a time
- concurrent vs. interleaving; multimodality

Task migratability
- passing responsibility for task execution between user and system
Principles of flexibility (2)

Substitutivity
- allowing equivalent values of input and output to be substituted for each other
- representation multiplicity; equal opportunity

Customizability
- modifiability of the user interface by user (adaptability) or system (adaptivity)
Principles of robustness (1)

Observability
- ability of user to evaluate the internal state of the system from its perceivable representation
- browsability; defaults; reachability; persistence; operation visibility

Recoverability
- ability of user to take corrective action once an error has been recognized
- reachability; forward/backward recovery; commensurate effort
Principles of robustness (2)

Responsiveness
- how the user perceives the rate of communication with the system
- Stability

Task conformance
- degree to which system services support all of the user's tasks
- task completeness; task adequacy
Standards

- set by national or international bodies to ensure compliance by a large community of designers; standards require sound underlying theory and slowly changing technology.

- hardware standards more common than software; high authority and low level of detail.

- ISO 9241 defines usability as effectiveness, efficiency and satisfaction with which users accomplish tasks.
Guidelines

- more suggestive and general
- many textbooks and reports full of guidelines
- abstract guidelines (principles) applicable during early life cycle activities
- detailed guidelines (style guides) applicable during later life cycle activities
- understanding justification for guidelines aids in resolving conflicts
Golden rules and heuristics

- “Broad brush” design rules
- Useful check list for good design
- Better design using these than using nothing!
- Different collections e.g.
  - Nielsen’s 10 Heuristics
  - Shneiderman’s 8 Golden Rules
  - Norman’s 7 Principles
Shneiderman’s 8 Golden Rules

1. Strive for consistency
2. Enable frequent users to use shortcuts
3. Offer informative feedback
4. Design dialogs to yield closure
5. Offer error prevention and simple error handling
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load
Norman’s 7 Principles

1. Use both knowledge in the world and knowledge in the head.
2. Simplify the structure of tasks.
3. Make things visible: bridge the gulfs of Execution and Evaluation.
4. Get the mappings right.
5. Exploit the power of constraints, both natural and artificial.
6. Design for error.
7. When all else fails, standardize.
Nielsen's Heuristics

1. Visibility of system status
2. Match between system & real world
3. User control and freedom
4. Consistency & standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility & efficiency of use
8. Minimalist design
9. Help error recovery
10. Help & documentation
1. Visibility of system status

Time Left: 00:00:19 searching database for matches

46%
What is “reasonable time”? 

- 0.1 sec: Feels immediate to the user. No additional feedback needed.
- 1.0 sec: Tolerable, but doesn’t feel immediate. Some feedback needed.
- 10 sec: Maximum duration for keeping user’s focus on the action.
- For longer delays, use % done progress bars.
2. Match between the system and the real world

**Socrates:** Please select command mode

**Student:** Please find an author named Octavia Butler.

**Socrates:** Invalid Folio command: please
3. User control and freedom

- Provide exits for mistaken choices
- Enable undo, redo
- Don’t force users to take a particular path
10. Help and documentation
We should wonder.....

- If this is a sensible heuristic set
  - Coverage
  - Uniqueness
  - Ease of use
4. Consistency and standards
5. Error prevention

People make errors.
Yet we can try to prevent them.

*How might you go about trying preventing errors?*
6. Recognition rather than recall

Ex: Can’t copy info from one window to another

Violates: *Minimize the users’ memory load*
7. Flexibility and efficiency of use

- Provide short cuts
- Enable macros
8. Aesthetic and minimalist design

![Screenshot of a form interface with fields for form title, heading, alternate text, background color, text color, background graphic, and a scrolling status bar message. The message reads: "WebMania 1.5b with Image Map Wizard is here!!"]
9. Help users recognize, diagnose, and recover from errors

SEGMENTATION VIOLATION! Error #13

ATTEMPT TO WRITE INTO READ-ONLY MEMORY!

Error #4: NOT A TYPEWRITER
Phases of a heuristic evaluation

1. Pre-evaluation training - give evaluators needed domain knowledge and information on the scenario
2. Evaluate interface independently
3. Rate each problem for severity
4. Aggregate results
5. Debrief: Report the results to the interface designers
Severity ratings

Each evaluator rates individually:
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

Consider both impact and frequency.
<table>
<thead>
<tr>
<th>EVALUATOR #</th>
<th>JL</th>
<th>AH</th>
<th>MH</th>
<th>DH</th>
<th>RvR</th>
<th>NH</th>
<th>MP</th>
<th>JC</th>
<th>DD</th>
<th>MC</th>
<th>SMC</th>
<th>PT</th>
<th>GA</th>
<th>SM</th>
<th>YB</th>
<th>AK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluators finding fewer problems</td>
<td>7</td>
<td>14</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>15</td>
<td>5</td>
<td>3</td>
<td>16</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Evaluators finding more problems</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROBLEM NUMBERS:</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>14</td>
<td>26</td>
<td>27</td>
<td>2</td>
<td>5</td>
<td>19</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>4</td>
<td>15</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>( # of evaluators finding each problem)</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
Styles of Heuristic evaluation

- Problems found by a single inspector
- Problems found by multiple inspectors
- Individuals vs. teams
- Goal or task?
- Structured or free exploration?
Problems found by a single inspector

- Average over six case studies
  - 35% of all usability problems;
  - 42% of the major problems
  - 32% of the minor problems

- Not great, but
  - finding some problems with one evaluator is much better than finding no problems with no evaluators!
Problems found by a single inspector

- Varies according to
  - difficulty of the interface being evaluated
  - the expertise of the inspectors

- Average problems found by:
  - novice evaluators - no usability expertise - 22%
  - regular specialists - expertise in usability - 41%
  - double specialists - experience in both usability and the particular kind of interface being evaluated - 60%
    - also find domain-related problems

- Tradeoff
  - novices poorer, but cheaper!
Problems found by a single inspector

- Evaluators miss both easy and hard problems
  - ‘best’ evaluators can miss easy problems
  - ‘worse’ evaluators can discover hard problems
Problems found by multiple evaluators

- 3-5 evaluators find 66-75% of usability problems
  - different people find different usability problems
  - only modest overlap between the sets of problems found
Problems found by multiple evaluators

- Where is the best cost/benefit?
Individuals vs. teams

- Nielsen
  - recommends individual evaluators inspect the interface alone

- Why?
  - evaluation is not influenced by others
  - independent and unbiased
  - greater variability in the kinds of errors found
  - no overhead required to organize group meetings
Self Guided vs. Scenario Exploration

- Self-guided
  - open-ended exploration
  - Not necessarily task-directed
  - good for exploring diverse aspects of the interface, and to follow potential pitfalls

- Scenarios
  - step through the interface using representative end user tasks
  - ensures problems identified in relevant portions of the interface
  - ensures that specific features of interest are evaluated
  - but limits the scope of the evaluation - problems can be missed
How useful are they?

- Inspection methods are *discount* methods for *practitioners*. They are not rigorous scientific methods.
  - All inspection methods are subjective.
  - No inspection method can compensate for inexperience or poor judgement.
  - Using multiple analysts results in an inter-subjective synthesis.
    - However, this also
      a) raises the false alarm rate, unless a voting system is applied
      b) reduces the hit rate if a voting system is applied!
  - Group synthesis of a prioritized problem list seems to be the most effective current practical approach.