Interface Design Rules
HCI Lecture 10

David Aspinall

Informatics, University of Edinburgh

23rd October 2007
Outline

Principles and Guidelines
   Learnability
   Flexibility
   Robustness

Other Guidelines
   Golden rules and heuristics
   HCI patterns

Standards
design rules have differing generality and operate at various levels. They:
- complement modelling and evaluation;
- encapsulate understanding and best practice;
- help us to design for maximum usability.
Types of Design Rules

- **principles**
  - abstract design rules
  - “an interface should be easy to navigate”

- **guidelines**
  - advice on how to achieve principle
  - may conflict; understanding theory helps resolve
  - “use colour to highlight links”

- **standards**
  - specific rules, measurable
  - “MondoDesktop links are RGB #1010D0”
Outline

**Principles and Guidelines**
- Learnability
- Flexibility
- Robustness

**Other Guidelines**
- Golden rules and heuristics
- HCI patterns

**Standards**
Usability Principles

- **Learnability**
  the ease with which new users can begin effective interaction and achieve maximal performance
Usability Principles

- **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
Usability Principles

- **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 to the user in determining successful achievement and assessment of goal-directed behaviour
Learnability (1): Predictability

**Predictability** — determinism and operation visibility

- System behaviour is observably deterministic:
  - Non-deterministic delays should be avoided
  - Operation effect determinable by interaction history

- Operation visibility:
  - User actions should be matched by a response
  - Affordance/logical constraints should be used to indicate available actions
Learnability (1): Predictability

**Predictability** — determinism and operation visibility

- System behaviour is observably deterministic:
  - *Non-deterministic delays should be avoided*
  - *Operation effect determinable by interaction history*
Learnability (1): Predictability

**Predictability** — determinism and operation visibility

- System behaviour is observably deterministic:
  - *Non-deterministic delays should be avoided*
  - *Operation effect determinable by interaction history*

- Operation visibility:
  - *user actions should be matched by a response*
  - *affordance/logical constraints should be used to indicate available actions*:

![Image of software interface with menu options]
Learnability (2): Synthesisability

**Synthesisability** — can assess effect of past actions

- Direct Manipulation interfaces promise *immediate honesty*
- Others have *eventual honesty*
- Command line interfaces are never honest:

```plaintext
Window  Edit  Options

dewar> ls
AdobeFnt.lst
Mail
Network Trash Folder
bin
documents
guitar
dewar> cd papers/
dewar>

lib
mail
mbox
ns_imap
nsmail
papers

public_html
research
software
teaching
```
Learnability (3): Familiarity

**Familiarity** — matching users’ expectations

- how prior knowledge applies to new system
  - *guessability* of the system
- knowledge of task and of other systems
- use of metaphor (e.g. tab-stops in word-processor)
- use of natural language syntax, affordances
  - *regions on the screen which denote buttons should be shaded to give a three-dimensional appearance*
Learnability (4): Consistency

**Consistency** — likeness in input/output behaviour arising from similar situations or task objectives
Learnability (4): Consistency

**Consistency** — likeness in input/output behaviour arising from similar situations or task objectives

- probably the most widely mentioned principle: “Be consistent!”
Learnability (4): Consistency

**Consistency** — likeness in input/output behaviour arising from similar situations or task objectives

- probably the most widely mentioned principle: “Be consistent!”
- challenge (and danger): consistency not self-contained
  - consistency within screens
  - consistency within applications
  - consistency within desktop
  - ...

Examples: consistent patterns in layout; same short-cut keys for similar action; same placement for recurrent menu options

Always place the Quit command as the last item in the leftmost menu.
Learnability (4): Consistency

**Consistency** — likeness in input/output behaviour arising from similar situations or task objectives

- probably the most widely mentioned principle: “Be consistent!”
- challenge (and danger): consistency not self-contained
  - consistency within screens
  - consistency within applications
  - consistency within desktop
  - ...
- Examples: consistent patterns in layout; same short-cut keys for similar action; same placement for recurrent menu options
  - *Always place the Quit command as the last item in the leftmost menu*
Learnability (5): Generalizability

**Generalizability** — extending specific interaction knowledge to new situations
Learnability (5): Generalizability

Generalizability — extending specific interaction knowledge to new situations

- helps give a predictive model of system for user
- a form of consistency
Learnability (5): Generalizability

**Generalizability** — extending specific interaction knowledge to new situations

- helps give a predictive model of system for user
- a form of consistency
- examples:
  - drawing circles → drawing ellipses
Learnability (5): Generalizability

**Generalizability** — extending specific interaction knowledge to new situations

- helps give a predictive model of system for user
- a form of consistency
- examples:
  - drawing circles → drawing ellipses
- UI standards and guidelines assist/enforce generalizability
  - *applications should offer the Cut/Copy/Paste operations whenever possible*
Flexibility (1) : Dialogue initiative

**Dialogue initiative** — who controls dialogue flow

- freedom from system imposed constraints on input dialogue
- user should be able to abandon, suspend or resume tasks at any point
- modal dialog boxes are system pre-emptive
- direct manipulation is user pre-emptive
- minimise system pre-emptive dialogue and maximise user pre-emptive dialogue
Flexibility (1) : Dialogue initiative

**Dialogue initiative** — who controls dialogue flow

- freedom from system imposed constraints on input dialogue
- *user should be able to abandon, suspend or resume tasks at any point*
Flexibility (1): Dialogue initiative

**Dialogue initiative** — who controls dialogue flow

- freedom from system imposed constraints on input dialogue
- *user should be able to abandon, suspend or resume tasks at any point*
- modal dialog boxes are *system pre-emptive*
- direct manipulation is *user pre-emptive*
- *minimise system pre-emptive dialogue and maximise user pre-emptive dialogue*
Flexibility (2): Multi-threading

**Multi-threading** — support simultaneous tasks
Flexibility (2): Multi-threading

Multi-threading — support simultaneous tasks

- concurrent vs. interleaving; multimodality
Flexibility (2): Multi-threading

**Multi-threading** — support simultaneous tasks

- concurrent vs. interleaving; multimodality
- *provide multiple task contexts*
Flexibility (3): Task migratability

**Task migratability** — how easily functions can be moved between user and system
**Flexibility (3): Task migratability**

**Task migratability** — how easily functions can be moved between user and system

- People get bored doing routine tasks and stop concentrating (Yerkes-Dodson Law)
- *automate routine tasks, but don’t fix function allocation*

<table>
<thead>
<tr>
<th>People are better at</th>
<th>Machines are better at</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detecting small sensory inputs</td>
<td>Responding quickly to signals</td>
</tr>
<tr>
<td>Improvising and using flexible procedures</td>
<td>Following procedures repeatedly and precisely</td>
</tr>
<tr>
<td>Reasoning inductively</td>
<td>Reasoning deductively</td>
</tr>
<tr>
<td>Selective information recall</td>
<td>Total information recall</td>
</tr>
<tr>
<td>Exercising judgement</td>
<td>Following orders</td>
</tr>
</tbody>
</table>
Flexibility (4): Substitutivity

**Substitutivity** — allowing equivalent values of input and output to be substituted for each other.
Flexibility (4): Substitutivity

**Substitutivity** — allowing equivalent values of input and output to be substituted for each other

- representation multiplicity; equal opportunity
- *don’t force users to refer to objects by name if they can point to them*
Flexibility (5): Customisability

**Customisability** — interface can be adapted to suit different needs
Flexibility (5): Customisability

**Customisability** — interface can be adapted to suit different needs

- modifiability of the user interface by user (adaptability) or system (adaptivity)
- *provide choice of methods; allow short-cuts; permit users to change features: deferred design.*
Robustness (1): Observability

**Observability** — user impression of system state

- Where am I? — immediate honesty wrt system state
- Where am I going? — operation predictability
- Where have I been? — synthesisability
- What can I do now? — predictability
Robustness (1): Observability

**Observability** — user impression of system state

- user should be able to evaluate the internal state of the system from its perceivable representation

- E.g., *Where*³ *What* of navigation:
  - Where am I? — immediate honesty wrt system state
  - Where am I going? — operation predictability
  - Where have I been? — synthesisability
  - What can I do now? — predictability
Robustness (2): Recoverability

**Recoverability** – support for undoing errors
Robustness (2): Recoverability

**Recoverability** – support for undoing errors

- ability of user to take corrective action once an error has been recognized
- reachability
  - *user should be able to undo back to any point*
- supported by reducing scope for making errors
  - *avoid free-form input where possible*
  - *validate input immediately, allowing correction*
- ... and ability of user to understand errors
  - *error messages should be concise, informative, specific, constructive*

<table>
<thead>
<tr>
<th>Poor:</th>
<th>Error 404: document not found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Better:</td>
<td>The requested URL <a href="http://www.foobaz.com/bar.html">http://www.foobaz.com/bar.html</a> could not be found</td>
</tr>
</tbody>
</table>
Robustness (3): Responsiveness

**Responsiveness** — feedback should be commensurate with action

- Sensitivity to delay depends on context
- *Echoing input* < 0.1 secs, *page turning* < 0.5 secs, *string search* < 4 secs

If delay is inevitable, provide reassurance: **time affordances**  
(Alex Paul Conn (1995))

A  acceptance
B  initiation and heartbeat
C  progress (fine-grained)
D  scope and remainder
E  exception
F  progress and completion
Robustness (4): Task conformance

**Task conformance** — degree to which the system supports the user’s tasks
Robustness (4): Task conformance

**Task conformance** — degree to which the system supports the user’s tasks

- Few general purpose commands, long methods, simple
- Many highly tuned commands, short methods, complex
- *identify core tasks; provide a command for each*
- But core task set grows over time; language is cluttered as lexicon expands
  - e.g., Unix command language, once a small set now has >700; 10% account for 90% of usage.
  - Microsoft Word command lexicon now includes text formatting, drawing, annotating, WWW related commands, etc.
Outline

Principles and Guidelines
  Learnability
  Flexibility
  Robustness

Other Guidelines
  Golden rules and heuristics
  HCI patterns

Standards
Golden rules and heuristics

- “Broad brush” design rules
- Useful check list for good design
- Better design using these than using nothing!

- 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

- Different collections e.g.:
  - Norman's 7 Principles (see Lecture 4)
  - Nielsen's 10 Heuristics (used in Heuristic Evaluation)
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.:
  - Norman’s 7 Principles (see Lecture 4)
  - Nielsen’s 10 Heuristics (used in Heuristic Evaluation)
  - Shneiderman’s 8 Golden Rules:
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.:
  - Norman’s 7 Principles (see Lecture 4)
  - Nielsen’s 10 Heuristics (used in *Heuristic Evaluation*)
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.:
  - Norman’s 7 Principles (see Lecture 4)
  - Nielsen’s 10 Heuristics (used in *Heuristic Evaluation*)
  - 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
HCI design patterns

- An approach to reusing knowledge about successful design solutions. Originated in architecture (Alexander).
- A pattern is an invariant solution to a recurrent problem within a specific context.
- Examples:
  - Light on Two Sides of Every Room (architecture)
  - Go back to a safe place (HCI)
- Patterns do not exist in isolation but are linked to other patterns in a pattern language which enables complete designs to be generated.
Outline

Principles and Guidelines
  Learnability
  Flexibility
  Robustness

Other Guidelines
  Golden rules and heuristics
  HCI patterns

Standards
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
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, *Ergonomics of Human System Interaction*, adopts traditional usability categories:
  - **effectiveness**: can you achieve what you want to?
  - **efficiency**: can you do it without wasting effort?
  - **satisfaction**: do you enjoy the process?
<table>
<thead>
<tr>
<th>Usability objective</th>
<th>Effectiveness measures</th>
<th>Efficiency measures</th>
<th>Satisfaction measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suitability for the task</td>
<td>Percentage of goals achieved</td>
<td>Time to complete a task</td>
<td>Rating scale for satisfaction</td>
</tr>
<tr>
<td>Appropriate for trained users</td>
<td>Number of power features used</td>
<td>Efficiency relative to expert user</td>
<td>Rating scale for ease of learning</td>
</tr>
<tr>
<td>Learnability</td>
<td>Percentage of functions learned</td>
<td>Time to learn criterion</td>
<td>Rating scale for ease of learning</td>
</tr>
<tr>
<td>Error tolerance</td>
<td>Percentage of errors corrected successfully</td>
<td>Time spent on correcting errors</td>
<td>Rating scale for error handling</td>
</tr>
</tbody>
</table>
References

Alex Paul Conn.
Time affordances: the time factor in diagnostic usability heuristics.

See also:
- Dix et al, Chapter 7.

Credits: some slides are due to Jon Oberlander, others are from web resources for Dix et al (see http://www.hcibook.com).