Design Principles and Heuristics

• general guidelines for developing intuitive and successful interfaces

• qualitative character

• platform-independent
– OS-independent
– hardware-independent

Can be used to
• Design the interface according to the “rules”
• Evaluate the interface in the heuristic evaluation
• Communicate with the developers

13 (heuristic) Design Principles

1. Simple and Natural Dialog, Let the User Develop a Mental Model
2. Make System Modes Easy to Recognize and Distinguish
3. Structure the Interface
4. Provide Shortcuts, Make the Interface Adaptable
5. Make Possible Actions Easily Visible
6. Be Consistent
7. Speak the User’s Language
8. Minimize the User’s Memory Load
9. Provide Feedback
10. Do not Surprise the User
11. Provide Clearly Marked Exits and Undo Functionality
12. Deal with Errors in a Positive Manner
13. Provide Help and Documentation

1) Simple and Natural Dialog, Mental Model

• use existing mental models (思维模式)
比如右上角退出，三指放大 ...

• present exactly the information the user needs
– less is more (less to learn, to get wrong, to distract, ...)
– information should appear in a natural order (cluster related information, order to match user’s expectations)
– remove or hide irrelevant or rarely needed information (competes with important information on the screen)
– remove (system) modes (e.g., edit mode vs. view mode) (减少模式的使用，以防用户忘记或混淆)

2) Modes Easy to Recognize & Distinguish

• modes: availability and meaning of commands (shortcuts, button clicks, etc.) depends on a mode

• system-controlled modes:
– the system initiates the mode or maintains the mode
– examples: select from the menu

• user-controlled modes:
– the user actively initiates and maintains the mode
– example: two-handed interaction on large displays

• avoid system-controlled modes if possible

3) Structure the Interface

• group similar functionality (把相似功能放在一起)
• structure the interface
• supports user's mental model (和用户的思维模式契合)
• new (and better) structure may initially perform worse (因为不适应)

4) Adaptable Interface and Shortcuts

Q: Explain why it is important to create adaptable interface and to provide shortcuts (at least 3)?

A: 1, People build the interface based on their own mental models (different environments and specific hardware).

2, Classes of users: novices, occasional users, experts. They have different expectations of the interface.

3, Different cognitive/visual abilities of users, different font sizes for visually impaired users, color schemes for users with color deficiency.

4, Shortcuts: expert users & frequent operations: mouse and keyboard accelerators.

5) Be Consistent

• consistent syntax of input
– shortcuts (比如win的复制是Ctrl-C，Ubuntu是Ctrl-Shift-C)

• consistent language and graphics
– same visual appearance across the system (布局一样)
– same information/controls in same location on all windows (比如win的关闭页面在右上，mac的在左上)

• consistent effects
– commands/actions have same effect in equivalent situations
→ predictability

6) Speak the User's Language

• use meaningful icons, abbreviations
– Meaningful icons
– Ctrl-S (abbreviation)

7) Minimize the User's Memory Load

• promote recognition over recall (别让用户记)
– menus, icons, dialogs vs. commands, options, formats
– relies on visibility of objects to the user (but less is more!)

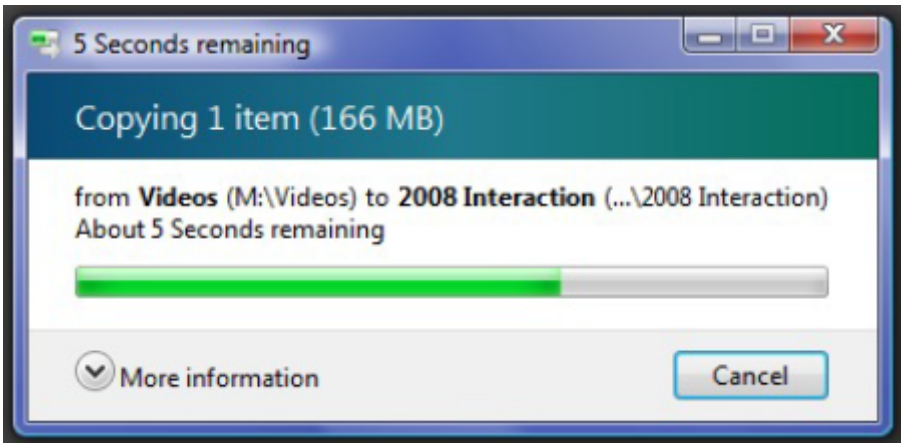
• give input format, example, and default (比如输入日期，直接出现日历让用户选)

• also show possible next steps/options or remind the user of shortcuts (help functionality) (比如"保存"图标，鼠标放上去会显示快捷键 和 名称)

8) Provide Feedback

• user should always be aware of what is going on

• continuously inform the user about
– what the program/interface is doing
– how it is interpreting the user's input



9) Do not Surprise the User

• programs should behave in a predictable way (mental model)
• results of actions should be foreseeable

9) Provide Clearly Marked Exits

• users do not like to feel trapped by the computer!
– interfaces should offer an easy way out of as many situations as possible

• strategies:
– cancel button (for dialogs waiting for user input)
– pause/continue (especially for lengthy operations)
– quit (for leaving the program at any time; ≠ cancel)

10) Provide Undo (and Redo) Functionality

• several steps of undo (and redo) should be possible
• not all actions need to be undone (scrolling?)
• visualizations of what could be undone possible (interaction history)
• if undo impossible: warn user!

11) Deal with Errors in a Positive Manner

• types of errors:
– mistakes: conscious deliberations lead to an error instead of the correct solution (比如做题做错)
– slips: unconscious behavior that gets misdirected route to a satisfying goal (比如打字拼错)

• designing for slips – general rules:
  – prevent slips before they occur
    • do not allow illegal actions or inputs
    • use masks and selection lists for data input
    • have as few different modes as possible
    • warn about unusual inputs
  – detect and correct slips when they do occur
    • self-correct error if possible (e.g., spell-check while typing)
  – user correction through feedback and undo
    • “let’s talk about it”: system initiates dialog to solve problem (e.g., compiler error)

12) Deal with Errors Positively: Provide Help

• provide meaningful error messages
• error messages in the user’s task language

13) Provide Help

Documentation

• many users do not read manuals
– want to spend time on task rather than reading about it

• usually used when users are in some kind of panic
– paper manuals often unavailable (locked/thrown away)
– online documentation better
– good search and lookup tools
– online help specific to current context

• sometimes used for quick reference
– syntax of actions, possibilities, list of shortcuts, etc.

Types of Help

• tutorials and/or getting started manuals

• reference manuals for lookup
– large ones for detailed reference
– short ones for quick lookup

• reminders, tooltips, contextsensitive help (e.g., syntax of function calls in programming)