design knowledge:
Explicit -- as rules
Implicit -- by patterns

Ch. 14.1 - 14.2.1 (pp. 644 – 660)

Explicit HCI design knowledge: design rules
Designing for maximum usability – the goal of interaction design
1. Principles of usability: General Understanding
2. Standards: Direction for design
3. Guidelines: Direction for design

types of design rules
1. principles
   1. abstract design rules
   2. low authority
   3. high generality
2. standards
   1. specific design rules
   2. high authority
   3. limited application
3. guidelines
   1. lower authority
   2. more general application

1 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

1.1 Principles of learnability

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

Synthesizability
- assessing the effect of past actions

Principles of learnability (ctd)

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
1.2 Principles of flexibility
Dialogue initiative
- freedom from system-imposed constraints on input dialogue
- Application- vs. user-preemptiveness
User-preemptiveness means that the interaction with the application is not restricted to any specific action determined by the application, but rather the user can choose freely from a variety of possible actions which to perform next. This is the case for WIMP.

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 (ctd)
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)

1.3 Principles of robustness
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 (=reasonable) effort

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

Principles of robustness (ctd)

2 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

Definition of usability
- Usability
  - The effectiveness, efficiency, and satisfaction with which specified users achieve specified goals in particular environments.
- Effectiveness
  - The accuracy and completeness with which specified users can achieve specified goals in particular environments.
- Efficiency
  - The resources expended in relation to the accuracy and completeness of goals achieved.
- Satisfaction
  - The comfort and acceptability of the work system to its users and other people affected by its use.
- Such a definition leads to design objectives and finally provides the means for explicit measurements for usability.
3 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 answering questions (or 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

3.1 Nielsen’s 10 Heuristics (1-5)

Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world
The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom
Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

http://www.useit.com/papers/heuristic/heuristic_list.html

3.1 Nielsen’s 10 Heuristics (6-10)

Recognition rather than recall
Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue in order to perform another.

Flexibility and efficiency of use
Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.

http://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html

3.2 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

http://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/shneidermanGoldenRules.html

3.3 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.

Implicit HCI design knowledge: design patterns

What is Functional Fixedness ("idea blocker") ... and how to overcome it?

Functional fixedness is a cognitive bias that limits a person to using an object only in the way it is traditionally used.

There are several examples of functional fixedness, one is the two-ropes problem (1931). Another is the Duncker (1945) candle problem.


The Two-Ropes Problem

The study of Maier (1931)
The author led his participants into a room with two ropes hanging from the ceiling. There were other tools in the room, like a hammer, pliers, and extension cords. This task often involves insight and is a demonstration of functional fixedness.
The task was to hold both ropes together.

Solution:
Bind the hammer or another heavy tool to one of the ropes and let them swing so that you can get the other rope.

Duncker (1945) candle problem.
Duncker (1945) candle problem.

Functional Fixedness in Software Design

- Rule of thumb for reusable SW: "make components generic".
- SW engineers tend to design in a way that solves the particular problem at hand, but do not easily see opportunities for reuse.
- Such solutions are "functionally fixed".
- Domain analysis prior to system design can help.
- Example: a more reusable QuickSort algorithm.