

HCI Principles

- Broader in scope than guidelines — “more fundamental, widely applicable, and enduring”
- On the other hand, this greater generality results in the need for more clarification, or individual interpretation by interaction designers
- The upshot is — before applying HCI principles, get to know the context of your users and applications very well first
- Many HCI gurus offer “their take” on design principles — frequently overlapping, with individual highlights

Principle: *Know Thy User*

- Corollary: if you think you know thy user, think again
- Diversity across multiple dimensions: age, gender, physical and cognitive abilities, education, culture or ethnicity, training, motivation, goals, personality
- Two types of user knowledge:
 1. Interface — knowledge of the technology that “presents” the system image
 2. Domain — knowledge of the real-world activities that your user interface seeks to accomplish

User Proficiency Profiles

	interface knowledge	domain/task knowledge
novice	little to none; shallow	little to none; shallow
first-time	little to none; shallow	knowledgeable
knowledgeable intermittent	some, but not specific	knowledgeable
expert frequent	expert	expert

Accommodating Multiple User Profiles

- Scope: are we designing for all profiles? (ideally yes) Or just a subset? (implies vertical applications)
- *Multi-layer, level-structured, or spiral* interfaces
 - Novices get “training wheels” — limited options, but also fewer opportunities for error
 - Increased proficiency enables increased functionality
 - Multiple layers include both software and documentation
- Not unlike progressing through a modern video game

Principle: Identify the Tasks

- a.k.a. “know thy domain”
- Dovetails with requirements and use-case analysis
- Task decomposition: what tasks are “atomic?” How are composite tasks put together?
- Task frequency: determines navigation structure, invocation methods
- Task selection: is “a better mousetrap” the same as “the kitchen sink?” — “*featuritis*” has been used to describe low usage-to-availability ratio

Principle: Choose an Interaction Style

An “interaction style” is the overall metaphor or paradigm that governs a user interface. The best user interfaces provide the best match between interaction style, user profiles, and tasks/domain. Of course, appropriate combinations are always acceptable.

- Direct manipulation: visual representation of domain
- Menu selection: sequential lists of choices
- Form fillin: labeled entry fields
- Command language: sequence of user directives
- Natural language: attempts to copy “H-H-I”

Principle: “Golden Rules”

HCI gurus provide individual takes on the most important elements of good user interfaces. Here are Shneiderman’s Eight Golden Rules:

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogs to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Nielsen’s Take: Ten Usability Heuristics

- Simple and natural dialog
- Speak the user’s language
- Minimize user memory load
- Consistency
- Feedback
- Clearly-marked exits
- Shortcuts
- Good error messages
- Prevent errors
- Help and documentation

Tognazzini's Take: Sixteen First Principles

Anticipation	Defaults	Human Interface Objects	Protect the User's Work
Autonomy	Efficiency of the User	Latency Reduction	Readability
Color Blindness	Explorable Interfaces	Learnability — Limit Tradeoffs	Track State
Consistency	Fitts's Law	Use of Metaphors	Visible Navigation

Discerning Patterns in the Rules

- Despite variations in phrasing and emphasis, certain common themes emerge among these (and other) sets of golden rules — this should give you an idea for prioritization and generality
- One way to reason about these rules objectively is to relate them to how they influence the five metrics of learnability, efficiency, memorability, errors, and subjective satisfaction
- It remains to be seen whether these rules will converge into the “one, true HCI rulebook”

“One Rule to Rule Them All” — **Prevent Errors**

- If there is any single golden rule that distinctly rises above the rest, it would be Shneiderman #5: Prevent Errors (a.k.a. Shneiderman #6, Nielsen #8 and #9, Tog #7 and #13)
- Consistency and feedback also enjoy multiple appearances in these lists, but they don't have the same bang-for-the-buck as error prevention
- Specific, positive, and constructive error messages
- Provide constant feedback on the state of the application — this suggests what can or cannot be done at a given time
- Correct actions: Understand conditions that may foster errors and either keep the user from performing erroneous actions or allow them to undo these actions
- Complete sequences: group multiple consistent steps into single, replicable composites
- Unit tests in software development build upon a similar idea of codifying sequences of steps

Principle: Integrating Automation vs. Control

- Sanders and McCormick (1993) suggest that we play to a human being's strengths as opposed to a machine's:
 - Avoid routine, tedious, and error-prone tasks — *automation*
 - Focus on making decisions, dealing with the unexpected, and planning for the future — *control*
- A corollary to control is *predictability* — we generally don't want the computer to “have a mind of its own”

- The FAA says it well: “improve system performance, without reducing human involvement” and “train users when to question automation” (2003)

- Automation vs. control grows in significance as *anthropomorphic* and *adaptive* user interfaces grow in popularity and sophistication
 - Microsoft's Office Assistants (or sometimes, Microsoft Office itself)
 - Assorted “bots” and pseudo natural-language interfaces (online help, search engines)
 - “Trained” spam filters
 - Amazon and others' “your store” or “your page” features