Heckel's Elements of Friendly Software Design

"In short, the software designer must learn to think like a communicator and to practice an artistic craft as well as an engineering one." (p. *xx*)

1. **Know Your Subject.**
2. **Know Your Audience.**
3. **Maintain the User's Interest.**
4. **Communicate Visually.**
5. **Lever the User's Knowledge.**
6. **Speak the User's Language.**
7. **Communicate with Metaphors.**
8. **Focus the User's Attention.**
9. **Anticipate Problems in the User's Perception.**
10. **If You Can't Communicate It, Don't Do It.**
11. **Reduce the User's Defensiveness.**
12. **Give the User Control.**
13. **Support the Problem-Solving Process.**
14. **Avoid Frustrating the User.**
15. **Help the User Cope.**
16. **Respond to the User's Actions.**
17. **Don't Let the User Focus on Mechanics.**
18. **Help the User to Crystallize his Thoughts.**
19. **Involve the User.**
20. **Communicate in Specifics, Not Generalities.**
21. **Orient the User in the World.**
22. **Structure the User's Interface.**
23. **Make Your Product Reliable.**
24. **Serve Both the Novice and the Experienced User.**
25. **Develop and Maintain User Rapport.**
26. **Consider the First Impression.**
27. **Build a Model in the User's Mind.**
28. **Make Your Design Simple...**
29. **But Not Too Simple.**
30. **You Need Vision.**

Ad 1. "Structured programming is a successful technique that has been used lately to develop better programs. However, one study shows that the quality of the customer interface is three times as important as structured programming to the success of a software project." (p. 22)

Ad 7. "Read the works of a good writer ..., and you will be surprised at the frequency with which images and analogies appear." (p. 36)

Ad 9. "Every good engineer, software or otherwise, expects his designs to operate under worst-case conditions and designs accordingly. ... One simple design technique is to have the program reinforce the *intent* of a command after the user uses that command" (p. 41)

"[T]he test of a user interface is to get *users* to try it." (p. 42, my emphasis)

Ad 12. "An important type of control we can give our users is to make their actions provisional." (p. 52)

Ad 14. "One common source of frustration is the user's manual. ... I view the user's manual as a software designer's list of failures. ... Another major source of frustration is waiting." (pp. 55–56)

Ad 22. "A fundamental challenge for engineers is to bring organization and structure to a design. A good structure gives the designer intellectual control over his design and helps him think about and communicate a design effectively. ... Probably the most important job that the designer must do—one that no one else can do—is to make the product an organic whole." (pp. 70–71)

Ad 24. "One of our problems as designers is that we quickly become experienced users and this causes us to lose our empathy with novices." (p. 75)

Ad 28. "Good design, like good writing, is simple and economical. Simple design makes the product easier to maintain and use. No magic formula will enable us to create simple designs; they are the result of creativity, false starts, and hard work. ... [F]requently we have to sacrifice the simplicity of internal design to make the user interface simple." (p. 84)

Ad 29. "There are two kinds of false simplicity. First, ... [w]hile the essence of a design should be simple, the details need not necessarily be. ... [S]econd ... [s]ometimes people think they are being simple when they are simplistic." (p. 86)

Ad 30. "The design of a good user interface does not come from knowing an extensive amount of literature on user interfaces, or worse yet, an extensive knowledge of software algorithms. It comes from having a vision of what the product should be." (p. 88)

Our Counterproductive Instincts

1. **We think logically, not visually.**
2. **We base our designs on our knowledge rather than the user's.**
3. **Our programs evaluate our user's actions.**
4. **We make our programs take control.**
5. **We think in generalities, not specifics.**
6. **We structure for internal organization.**
7. **We strive for a program's internal simplicity.**
8. **Our knowledge constrains our vision.**

Innovation Acceptance

"Everett Rogers discovered that people's *perception* of five factors associated with an innovation determines how fast they will accept it." (p. 91, my emphasis)

1. **Ease of use.**
2. **Relative advantage:** "The most important factor ... is the advantage it has over the existing way of doing things." (p. 91)
3. **Compatibility with the user's environment:** "Since basic human nature resists change, our designs, no matter how innovative they are, should minimize the changes the users perceive they need to make in how they do things." (p. 92)
4. **Trialability.**
5. **Observability:** "The more visible an innovation is the more quickly it will be accepted. ... The designer can make a product observable in several ways. First, he can make the product so that it is interesting at the point of sale. Second, he can design it so that it is naturally used where people see it. Finally, it can make several of its features visible to the user and thus stimulate their use." (p. 94)

Observing People

"If we are going to design software for people, we have to observe how people, including ourselves, use products and what they use them for. We should also observe how other designers have solved their user interface problems." (p. 105)

Patent Protection

"[S]oftware is what makes computers useful. ... The thesis of this book is that writing friendly software is a communications craft. Consistent with this position, the software developer should get the same protection the writer, playwright, or other communicator gets." (pp. 217, 220)

These quotations have been taken from: Paul Heckel, *The Elements of Friendly Software Design*, Second Edition, SYBEX Inc., 1991.

Tom Verhoeff 3 July 1992