The UNIVERSITY of VERMONT

Benjamin Fenster
CS-296 (Damon)
15 March 2005

Design

I. Goals
   a. Easy to use, efficient, aesthetic
   b. Are there special devices?
   c. Are there issues with the target audience?

II. Steps
   a. Task analysis. What will the user be doing?
   b. Metaphor development
   c. Affordances
   d. Look and Feel – intended overall impression
   e. Dialog Design – What's the sequence of steps?
   f. Information Presentation – How will it be presented (sometimes easy, sometimes hard)
   g. Layout

III. Task Analysis
   a. What does the user want to accomplish?
   b. What do they know?
   c. What do they need to know?
   d. How is it done now? Use that model if possible.
   e. User Analysis
      i. What can they do? What skills do they have?
      ii. What *can't* they do? What are they unwilling to do (e.g. doctors feeling things are beneath them)?
   f. Tasks
      i. Point accomplishments. One focused short-term goal.
      ii. Distinct from the *features*. Those are action oriented (what can you do). Here we're describing what the user *wants* to do.
   g. Use Cases
      i. Structured document
      ii. There's some graphical notation, but it's not really helpful.
      iii. Includes who, what, why, when
      iv. Have one use case for each task.
      v. Always written from the perspective of the *goal*, not the *solution*. You'll say "confirm the meeting," not "click OK."
   h. Sources
      i. Documentation
         1. Tells what other products already do.
         2. It's easy to find manuals; it's usually easy to interpret tasks in them.
      ii. Guessing
         1. Can try to guess about what the users want.
         2. You'll probably be really bad at it.
      iii. Ask the Users
         1. Make sure you're talking to the real users, not a manager who's just passing work down to a secretary and who doesn't really know what's needed.
         2. May not get good answers about what users want, and it's hard to figure out what questions to ask.
      iv. Observe Users
         1. Observe everything the user does.
         2. You'll see two things happening and know they should be done together.
         3. Customer behavior while being watched isn't always "normal behavior"
      v. Task Decomposition
         1. You know the larger problem.
         2. Decompose it into smaller goals.

IV. Metaphor

a. A good metaphor improves both learning and efficiency.
b. Can encourage the user to take some actions faster, for example.
c. Obviously user needs to buy into the metaphor.
d. Want *everything* to fit the metaphor.
e. It's *really* hard to get a great metaphor. "Okay" metaphors happen all the time.
f. Where to get ideas?
    i. Existing processes
    ii. Physical system (even something that's not real, or doesn't perfectly follow the laws of physics)
    iii. Imagination

V. Affordances
a. By itself, an affordance doesn't do anything. It just gives a clue about how to do something.
b. Can be a decoration (painted handle on the fridge). Can be a layout association.
c. Shapes (square = checkbox, circle = radio button) can be affordances.
d. Introduce the metaphor (show the way the physical thing works – get in the mindset)
e. Foreshadowing – what's coming next
f. Use as many affordances as you can without overwhelming the user.
g. User testing will show a lack of affordances: if people get confused, they need more hints

VI. Look and Feel
a. Be consistent with the platform, first of all.
b. Second, the look and feel id dictated by the design goal (simple vs. "ooh ahh.")
c. It's also dictated by time constraints.

VII. Dialog
a. Who does what and when?
b. Not too much detail – don't care about whether there are dashes in SSNs.
c. People don't want to watch the computer. People don't want to just type.
d. People want to be having a dialog. Good interfaces promote and facilitate a dialog.
e. Dialogs (*any* dialogs) can be described at three levels
    i. Semantics: What is the meaning?
    ii. Syntax: What is it legal to say and when?
    iii. Lexical: Low-level syntax (*words*)
f. In HCI
    i. Lexical means what controls you offer
    ii. Syntax is defined by the dialog itself.
    iii. Semantics describes what a control does.
g. Syntax should map easily to semantics. Otherwise you've got a very modal application.
h. There's no clear boundary between lexical and syntactical
i. Specific Dialogs
    i. Who's allowed to say what when?
    ii. Among people there are rules about who's turn it is, who's allowed to interrupt and when, what's appropriate to say.
    iii. Ideally *style* is orthogonal to *dialog*.
j. Notation
    i. Need to communicate the dialog to yourself and others (i.e. users)
    ii. There are many, many notations
    iii. Simple Text
        1. Human: Xxxxx Computer: Yyyyy
        2. May have variables "Do you, ____, take ____, to be your lawfully wedded husband?"
        3. May also have decision points. "How do you plead?" "Guilty / Not Guilty"
        4. It's easy to write and usually easy to read, but terrible at describing choices. Even the "Guilty / Not Guilty" example isn't completely clear.
    iv. Grammars
        1. Good at explaining choices. Gives a precise indication of what and when.

2. It's hard to see who does what at the high level. Can't see the forest through the trees.

v. Finite State Machines
   1. States say where you are in the program
   2. Arrows tell user actions.
   3. You get lots of arrows and lots of states very easily.
   4. Common exits and common sequences (e.g. "help", which does essentially the same thing but returns to a different point depending on where you started) are hard to model.
   5. It's still pretty good, despite the complexity.

vi. State Charts
   1. Can group states together.
   2. This is a huge improvement over Finite State Machines.

vii. Formal Notations
   1. State Machines, State Charts, and Grammars are all formal notations. There are other formal notations we won't cover.
   2. Automatic Generation. By building a formal description you can automatically build the interface code that does lots of the work.

k. Designing Dialogs
   i. Refining Tasks
      1. Have to get down to single actions (typing in a text field).
      2. Some tasks get dropped (like the user thinking about something)
   ii. Add Decision Points. End up with a "tree" structure. It's not really a tree since it will contain cycles and merges, but it should seem a lot like one.
   iii. Define Ordering.
      1. Is the ordering required? Is it just what's comfortable? Consistent? Know why you're doing it that way!
      2. Remember the user wants maximum control.

l. Doing Design
   i. Build the complete flow for each task
   ii. Check parallelism (are things in the same order on multiple tasks?)
   iii. Evaluate
   iv. State Machines / State Charts make it easy to evaluate completeness. You should have the same set of arrows coming from every state. So if something's illegal, indicate that explicitly.
   v. Check consistency across tasks (of computer responses)
   vi. Check "reachability"
      1. Make sure the user can't back into some corner
      2. Look for isolated-looking islands on the state chart
      3. (Example: Computer says, "Floppy not formatted" but won't allow format)
      4. If the system provides the only way out, make sure it's possible for the code to generate that way out!
   vii. Neighborhood. Related states should be easily reachable (e.g. setting font size, font face). What's the shortest path from X to Y?

VIII. Information Presentation
a. Can enrich or hurt the metaphor.
b. Restricted by human perception
c. Kinds
   i. Continuous Numerical (scientific data, graph it)
   ii. Discrete Numerical (charts)
   iii. Sequential (charts, calendar)
   iv. Active, Small amounts (e.g. microphone level at this moment – have an individual indicator for each datum).
   v. Categorization, variation over time, relationships
d. Resources
   i. Edward Tufte ("Grand Poobah" of information presentation)

ii. Robert Spence

e. Classics: Bar charts, pie charts, …

f. Variables
   i. Size (line thickness, radius of dots)
   ii. Shape
   iii. Color
   iv. Texture
   v. Position
   vi. Connections
   vii. Text (labels)

g. Potential Problems
   i. Color and texture has perceptual problems. Use different *shades* too.
   ii. Shape has limited information density.
   iii. Lines can get easily overcrowded (think of a BlueJ UML diagram)
   iv. Size
      1. Can be badly misinterpreted (area especially). Increasing edges of a square 40% doubles the area. People don't see it that way though.
      2. Don't "mislead" by more than 5% bigger or smaller than the size should really be (based on other objects). You get some artistic license, but if it's off by too much it's basically lying.

h. Guidelines
   i. Data-Ink. You want most of the ink (non-background pixels on the computer) to be representing data as opposed to art.
   ii. Don't go overboard and *just* show data points on a scatter plot, but be aware of the data-ink ratio.
   iii. "Ducks" – don't change the *form* in order to make it look pretty.

i. Changing View
   i. When the data don't fit on one page or screen, or if the user can interact with the data, you need to be more active than just displaying.
   ii. Scrolling
      1. Everything from simple 1D scrolling up to elaborate 3D rendering with data gloves and a wraparound screen.
      2. In traditional scrolling, it's very easy to lose context (especially when you can jump to an arbitrary point).
      3. You want some "big picture" in addition to the detail, like the map in the corner of a gaming screen.
      4. Can enlarge the detail section (magnifying glass of some sort) or shrink the context (3D effect of having non-focused data trailing into the screen)
   iii. Pan & Zoom
      1. Panning is just scrolling
      2. In some cases zoom just changes the magnification.
      3. Zoom can also mean adding data without changing the magnification (e.g. adding details to a network diagram box in MS Project)

j. Information in Controls
   i. Classic example: the size of the scrollbar changes to reflect the document size
   ii. Another classic: Mail programs bold mail folder names or change the icon to indicate that they contain new mail.
   iii. Don't want to go overboard and have stuff moving around too much

k. Active Information
   i. Data *is* the control
   ii. Drag things on a map, for example.

IX. Interaction Design

a. "I'll use a TextBox, not a ComboBox."

b. Map each user action to a control on the screen. Most of these are trivial.

c. Style (WIMP, …) does play a huge part now.

d. WIMP Controls

i. Text Field
    1. One line or many? Scrolling? Validation? Initial value?
    2. Need to think about all these things.
ii. Button: Picture or text? When (if ever) is it the default?
iii. Menus: How deep, how wide, how are items grouped?
iv. Click / Select: How much is selected? What if you click nothing? Shift- or control-click? Lots of combinations to consider!

e. Custom Controls
    i. May support the metaphor or give more subtle control to the user.
    ii. If it's unusual, remember it'll take user some time to learn.
    iii. Alphabetic Slider:
        1. |______/_____|
           A B C D E F G H I J K L …
        2. Conveys information about how many elements are under each letter and precisely where the display is currently located (toward the beginning of the Cs).

f. Controls can appear / disappear as you need them in software. In consumer electronics, on the other hand, you have a certain number of buttons. It costs more money to add more buttons, and you can't change them in mid-use.

g. Device Controls
    i. Buttons, rocker controls, slider.
    ii. Toggles (switches), touch pad
    iii. Rotary switch (n-way selector)
    iv. Continuous Rotary Knob
    v. Can't disable controls either, so they must always do *something*
    vi.

X. Layout

a. You now know what controls to use. Where do you want to put them?

b. Grouping
    i. This is the most important thing!
    ii. Break unrelated / separate chunks apart
    iii. Breaking across pages is generally bad
        1. Copy over anything needed from previous screens if you have to insert a page break.
        2. This limits the user's control.
        3. It's also hard for the user to retain context.
    iv. On the other hand, page breaks are good sometimes
        1. When you *want* the user to see a new context.
        2. This facilitates high-level chunking

c. Layout Within Pages
    i. Grouping, grouping, grouping.
    ii. Remember six degree field of vision (about two inches on the screen)

d. Four Guidelines
    i. Proximity. Be explicit about when things go together and when they're apart.
    ii. Alignment
        1. Align elements where it's reasonable.
        2. When centering, vary the length of each line dramatically. If lines of text are a similar length and centered, the edges just look ragged.
    iii. Consistency
        1. Use the same basic structure on every page.
        2. Also use the same font, colors, et cetera.
        3. Use "equivalent" graphics, labels, et cetera. If you have a black and white clip art as the "logo" for one page, don't use a full color photograph for the "logo" on another page.
    iv. Contrast
        1. When pages *are* different, make it a striking difference.

2. Differentiate headers from text. Don't just use a 2 point font difference.

e. Density
   i. Greater density gets more information on a page, but makes it harder to read.
   ii. It may make perfect sense to cram tons of information together if people can get used to reading it that way.

f. Tullis Metrics
   i. Overall density: % of page filled with data (thinking originally about 80 x 20 ASCII displays, but this can be applied to graphical displays too)
   ii. Local density: % of five-degree area filled with data
   iii. Complexity: (scary calculation) How many *different* things are on any line across the screen?
   iv. Separation: Twice the average distance between characters. The point is high-density text needs more separation to make it identifiable as a group
   v. A group is a block of characters within "separation" of each other. Then Group Size is the average visual *angle* of groups

g. More Guidelines
   i. Optimize for the order in which you expect people to use the controls most often.
   ii. If there are two viable orders ABCD and ACBD then you might arrange the controls: AB / CD
   iii. Start at the top-left and then put OK in the bottom-right.
   iv. The first few things in a clump should make the purpose of the clump obvious.
   v. Put required things on top and optimal things on the bottom.
   vi. Affordances
      1. Layout shows some affordances about sequences.
      2. Include additional affordances where possible.
      3. A progress bar ("Buy → Checkout → Ship") is a good example.
   vii. Multiple Windows.
      1. Not talking about multiple *documents*, but popup windows with additional information within one dialog.
      2. Close child windows when the parent closes.
      3. Tie related windows together visually.
   viii. Personal Control
      1. Give the user as much control as possible.
      2. Remember on websites the user *already* has control so you'd better account for that! Don't convey too much information with font, for example, since the user can change the font!

h. What do the Webby winners have in common?
   i. Short links (three words or fewer)
   ii. Sans Serif Fonts. Georgia, not Times
   iii. When using Serif, use Verdana, not Arial.
   iv. Color
      1. Four at once looks about right (per page / screen)
      2. Up to about seven for the whole application
      3. Design for monochrome first, then add colors for aesthetics, emphasis, and contrast.

XI. Platform Guidelines
   a. These aren't standards or APIs, they're just guidelines for how everything on a platform should look.
   b. Apple started this with the Apple Human Interface Guidelines.
   c. The guidelines for Aqua are about 300 pages long.
   d. Apple used to endorse products that followed all the guidelines.
   e. Private Guidelines
      i. Companies define these for their own products.
      ii. What makes this look like an Adobe product, for example?
      iii. Called a Style Sheet sometimes (a magazine term)

    iv. Not usually publicly available.
f. Guidelines vs. Principles
    i. Guidelines are very specific (“default button highlighted in light blue”)
    ii. Principles are more general.