Inspiration

Cooper books
Articles
Newsletter
Concept projects
Book reviews

The Myth of Metaphor

by Alan Cooper, Chairman & Founder

June 1995

Originally Published in Visual Basic Programmer's Journal

Software designers often speak of "finding the right metaphor" upon which to base their interface design. They imagine that rendering their user interface in images of familiar objects from the real world will provide a pipeline to automatic learning by their users. So they render their user interface as an office filled with desks, file cabinets, telephones and address books, or as a pad of paper or a street of buildings in the hope of creating a program with breakthrough ease-of-learning. And if you search for that magic metaphor, you will be in august company. Some of the best and brightest designers in the interface world put metaphor selection as one of their first and most important tasks.

But by searching for that magic metaphor you will be making one of the biggest mistakes in user interface design. Searching for that guiding metaphor is like searching for the correct steam engine to power your airplane, or searching for a good dinosaur on which to ride to work.

I think basing a user interface design on a metaphor is not only unhelpful but can often be quite harmful. The idea that good user interface design is based on metaphors is one of the most insidious of the many myths that permeate the software community.

Metaphors offer a tiny boost in learnability to first time users at

tremendous cost. The biggest problem is that by representing old technology, metaphors firmly nail our conceptual feet to the ground, forever limiting the power of our software. They have a host of other problems as well, including the simple fact that there aren't enough metaphors to go around, they don't scale well, and the ability of users to recognize them is questionable. Confusing the issue is the problem that most of what we consider metaphoric interface isn't.

The three interface paradigms

I think that there are three dominant paradigms in software user interfaces. I call these three the technology paradigm, the metaphor paradigm, and the idiomatic paradigm. The technology paradigm is based on understanding how things work, a difficult proposition. The metaphor paradigm is based on intuiting how things work, a problematic method. The idiomatic paradigm is based on learning how to accomplish things, a natural, human process.

In general, we have been progressing from technology through metaphor and are just now becoming aware of idiomatic design. Although there is ample evidence of all three in contemporary software, the metaphor paradigm is the only one that has been popularized, so we pay it lots of lip service and, all too often, hamper the creation of really good interfaces by following its false trail.

The technology paradigm

The technology paradigm of user interface is simple and incredibly widespread in the computer industry. It merely means that the interface is expressed in terms of its construction, of how it was built. In order to successfully use it, the user must understand how the software works.

There was a movement in building architecture in the 1960's called Metabolist, and its influence is still evident. In metabolist architecture, the elevator shafts, air conditioning ducts, cable runs, steel beams and other construction impedimenta are left uncovered and readily visible from both inside and out. The muscles, bones and sinews of a building are exposed-even emphasized-without any hint of modesty. The idea being that the building is a machine for living and its form should

follow its implementation details. The overwhelming majority of software programs today are metabolist in that they show us without any hint of shame precisely how they are built: One button per function, one function per module of code, commands and processes that precisely echo the internal data structures and algorithms.

We can see how a technology program ticks merely by learning how to run it; The problem is that the reverse is also true: We must learn how it ticks in order to run it. Engineers want to know how things work, and the technology paradigm is very satisfying to them, which, of course, is why so much of our software follows it. Engineers prefer to see all of the gears and levers and valves because it allows them to understand what is going on inside the machine. That those artifacts needlessly complicate the interface seems a small price to pay. Although engineers want to understand the inner workings, most users don't and lack the time if they do. They'd much rather be successful than be knowledgeable, a state that is often hard for engineers to understand.

The metaphor paradigm

In the 1970s, the modern graphical user interface was invented at Xerox Palo Alto Research Center (PARC). It has swept the industry, but what, exactly, is it? The GUI, as defined by PARC consisted of many things: Windows, buttons, mice, icons, metaphors, pulldown menus. Some of these are good and some are not so good, but they have all achieved a kind of holy stature in the industry by association with the empirical superiority of the ensemble. In particular, the idea that metaphors are a firm foundation for user interface design is a very misleading proposition. It's like worshipping 5.25" floppy diskettes because so much good software once came on them.

The first commercially successful implementation of the PARC GUI was the Apple Macintosh, with its desktop metaphor, wastebasket metaphor, overlapping sheets of paper metaphor and file folder metaphor. The success of the Mac wasn't because of these metaphors but because it was the first computer that defined a tightly restricted vocabulary for communicating with users based on a very small set of mouse actions. The metaphors were just nice paintings on the walls of a well-designed house.

Metaphors don't scale very well. A metaphor that works well for a simple process in a simple program will often fail to work well as that process grows in size or complexity. Icons for files was a good idea when computers had floppies or 10 megabyte hard disks. In the days of gigabyte hard disks and thousands of files, icons can get pretty clumsy. We understand metaphors by intuition. In user interfaces, we grasp the meaning of the metaphoric control because we mentally connect it with some other process or thing we have already invested time and effort into learning. The great strength of this method is its efficiency, taking advantage of the awesome power of the human mind to make inferences, something that CPUs are incapable of. The weakness of this method is that it depends on the creaky, cantankerous, idiosyncratic human mind, that may not have the requisite language, knowledge or inferential power necessary to make the connection. Metaphors are not dependable the way that understanding is. Sometimes the magic works, sometimes it doesn't.

The intuition of the metaphor paradigm takes place without understanding the mechanics of the software, so it is a step forward from the technology paradigm, but its power and usefulness has been inflated to unrealistic proportions. Webster defines intuition as "the power or faculty of attaining to direct knowledge or cognition without evident rational thought or inference." Wow! No thinking involved. It is silly to imagine that we can base good user interface design on a kind of mental magic that thumbs its nose at thinking. We intuit things by mentally comparing them with what we have already learned. You instantly intuit how to work a wastebasket icon because you once invested the effort to learn how to work a real wastebasket, preparing your mind to make the connection years later. But you didn't intuit how to use the original wastebasket. It was just extremely easy to learn. Which brings us to the idiomatic paradigm, which is based on the fact that the human mind is an incredibly powerful learning machine, and that learning isn't hard for us.

The idiomatic paradigm

This third method of user interface design solves the problems of both of the previous two. I call it idiomatic because it is based on the way we learn and use idioms, or figures of speech, like "beat around the

bush" or "cool." They are easily understood but not in the same way metaphors are. There is no bush and nobody is beating anything. We understand the idiom because we have learned it and because it is distinctive. Pretty simple, huh? This is where the human mind is really outstanding, mastering learning and remembering idioms very easily without having to depend on comparing them to known situations or understanding how they work. It has to, because most idioms don't have any metaphoric meaning at all. Most of the controls on a GUI interface are idioms. Splitters, winders, comboboxes and scrollbars are things we learn idiomatically rather than intuit metaphorically.

We tend to think that learning is hard because of the conditioning we have from the technology paradigm. Those old user interfaces were very hard to learn because you also had to understand how they worked. Most of what we know we learn without understanding; things like faces, social interactions, attitudes, the arrangement of rooms and furniture in our houses and offices. We don't "understand" why someone's face is composed the way it is, but we "know" their face. We recognize it because we have looked at it and memorized it, and it wasn't that difficult.

The familiar mouse is not metaphoric of anything but rather is learned idiomatically. That scene in Star Trek IV where Scotty returns to twentieth-century Earth and tries to speak into a mouse is one of the few parts of that movie that is not fiction. There is nothing about the mouse that indicates its purpose or use, nor is it comparable to anything else in our experience, so learning it is not intuitive. However, learning to point at things with a mouse is incredibly easy. Someone probably spent all of three seconds showing it to you your first time, and you mastered it from that instant on. We don't know or care how mice work and yet we can operate them just fine. That is idiomatic learning.

The key observation about idioms is that although they must be learned, good ones only need to be learned once. It is quite easy to learn idioms like "cool" or "politically correct" or "kick the bucket" or "the lights are on but nobody's home" or "in a pickle" or "inside the beltway" or "take the red-eye" or "grunge." The human mind is capable of picking up an idiom like one of the above from a single

hearing. It is similarly easy to learn idioms like checkboxes, radiobuttons, pushbuttons, close boxes, pulldown menus, buttcons, tabs, comboboxes, keyboards, mice and pens.

This idea of taking a simple action or symbol and imbuing it with meaning is familiar to marketing professionals. Synthesizing idioms is the essence of product branding, whereby a company takes a product or company name and imbues it with a desired meaning. Tylenol is a meaningless word, an idiom, but the McNeil company has spent millions to make you associate that word with safe, simple, trustworthy pain relief. Of course, idioms are visual, too. The golden arches of MacDonalds, the three diamonds of Mitsubishi, the five interlocking rings of the Olympics, even Microsoft's flying window are non-metaphoric idioms that are instantly recognizable and imbued with common meaning.

Ironically, much of the familiar GUI baggage often thought to be metaphoric is actually idiomatic. Such artifacts as window close boxes, resizable windows, infinitely nested file folders and clicking and dragging are non-metaphoric operations-they have no parallel in the real world. They derive their strength only from their easy idiomatic learnability.

The showstoppers

If we depend on finding metaphors to create user interfaces, we encounter the several minor problems already mentioned, but there are two more major problems: metaphors are hard to find and they constrict our thinking.

It may be easy to discover visual metaphors for physical objects like printers and documents. It can be difficult or impossible to find metaphors for processes, relationships, services and transformations, the most frequent use of software. It can be extremely daunting to find a useful visual metaphor for buying a ticket, changing channels, purchasing an item, finding a reference, setting a format, rotating a tool or changing resolution, yet these operations are precisely the type we find in software most frequently.

The most insidious problem with metaphors, the showstopper, comes

from tying our interfaces to mechanical age artifacts. It is easy to intuit how to use the clipboard, for example, because it is a metaphor. But to adhere to the clipboard metaphor, the facility is incredibly weak. It won't hold more than one thing, it doesn't have a memory of what it held before, it can't identify where the images came from, it can't show you thumbnails of what it holds and it doesn't save its contents from run to run. All of these actions are non-metaphoric and would have to be learned. Following the metaphor gives users a momentary boost the first time they use the clipboard but it costs them greatly ever after in the arbitrary weakness of the facility.

Another really outrageous example is MagicCap, the new communications interface from General Magic. It relies exclusively on metaphor for every aspect of its interface. You metaphorically walk down a street lined with buildings representing services. You enter a building to begin a service, represented as a walk down a hallway lined with doors representing functions. The heavy reliance on metaphor means that you can intuit the basic functioning of the software, but the downside is that the metaphor restricts all navigation to a very rudimentary, linear path. You must go back out onto the street to find another service. In the physical world this is normal but in the world of software there is no reason to force the user into those clumsy old methods. Why not abandon the slavish devotion to metaphor and provide the user with services they can't get out on the street?

Don't get me wrong, there is nothing bad about using a metaphor if one happens to fit the situation. Look! Here's a twenty-dollar bill lying on the sidewalk. Of course I'll pick it up; I'd be a fool not to! But then, I'd be a bigger fool if I decided to make my living finding misplaced twenty-dollar bills. Metaphors are like that: use 'em if you find 'em, but don't bend your interface to fit some arbitrary metaphoric standard.

It may seem clever to have your dial-up service represented by a picture of a telephone sitting on a desk, but it actually imprisons you in a bad design. The original makers of the telephone would have been ecstatic if they could have created one that let you call your friends just by pointing to pictures of them. They couldn't because they were restricted by the dreary realities of electrical circuits and

bakelite moldings. On the other hand, today we have the luxury of rendering communications any way we please-showing pictures of our friends is completely reasonable-yet we insist that on holding communications back with little pictures of obsolete techology.

The temptation is irresistable to stretch the metaphor well beyond just recognizing the function: That little desktop telephone also lets you "dial" with buttons just like those on our desktop telephones. We see software that has "address books" of phone numbers just like those in our pockets and purses. Wouldn't it be better to go beyond these confining technologies and deliver some of the real power of the computer? Why can't our communications devices allow multiple connections or make connections by organization or affiliation, or just abandon phone numbers altogether?

The future of user interface design will be idiomatic, where we depend of the natural ability of humans to learn easily and quickly as long as we don't force them to understand how and why. There is an infinity of idioms waiting to be invented, rather than a limited set of metaphors waiting to be discovered. Metaphors give first-timers a penny's worth of value but cost them dollars worth of problems as they continue to use the software. It is always better to design idiomatically, only using the occasional metaphor when one falls in our lap.

back to previous page

back to top

home company services news inspiration contact us