# Working with Interface Metaphors

**Thomas D. Erickson**
*Advanced Technology Group*
*Apple Computer, Inc.*

"METAPHOR," someone recently said to me, "seems to be the holy grail at Apple." It's true. Just about everyone at Apple knows the phrase "desktop metaphor" and fervently believes that a good metaphor is essential to an easy-to-use human interface. But just as the grail proved to be elusive, so is the knowledge of how metaphor really works.

The goal of this chapter is to provide designers with a deeper understanding of what metaphor is, and how to use it when designing an interface. First, we look at some of the characteristics of metaphor in language. These characteristics provide insights on how metaphor works in an interface. Next we look at an example of a poorly chosen interface metaphor and how it decreases the usability of the system. Finally, a design example is used to illustrate a method for coming up with interface metaphors and to present some rules of thumb for evaluating their value.

## The Ubiquity and Invisibility of Metaphor

Many people think of metaphor as a flowery sort of language found chiefly in poetry and bad novels. This is not correct. Metaphor is an integral part of our language and thought. It appears not only in poetry and novels, but in our everyday speech about common matters. Mostly we don't notice; metaphor is such a constant part of our speech and thought that it is invisible.

The ubiquity of metaphor in language is convincingly demonstrated in

the delightful book, *Metaphors We Live By* [Lakoff and Johnson, 1980]. The authors show that many of our basic concepts are based on metaphors. For example, without any intention of being poetic or fanciful, we speak of argument as though it is war. Arguments have *sides* that can be *defended* and *attacked*. Facts can be *marshaled* to support one's *position; strategies* can be employed. If a position is *indefensible*, one can *retreat* from it. Arguments can have *weak points*—they can even be *destroyed*; arguments can be *right on target*; arguments can be *shot down*. There is a whole web of concrete military language that we use to describe the rather abstract process of having an argument. It's also important to note that we don't just talk about argument as though it were war: being in an argument feels like a conflict; when we *lose* an argument, we feel bad.

The metaphorical way in which we talk and think about argument is the rule, not the exception. For example, the goal of this chapter is to provide a *deeper* understanding of metaphor. (Rather than just providing a *surface* treatment or *getting our feet wet*, I'd really like to *get into* the topic. Yet metaphor contains unexpected *depths*. Although we must avoid *getting in over our heads*; still, it would be nice to *plunge in* and *get to the bottom* of things.) In the next sentence, I say that we're going to *look at* metaphor as it's used in language. You *see*, we often speak of understanding an idea as seeing an object. Thus we may want to take a *closer look* at something, *shed some more light* on it, or *approach it from a different direction* so as to get a new *perspective* on it.

A word that is used in a metaphorical way is usually just the tip of the iceberg. A metaphor is an invisible web of terms and associations that underlies the way we speak and think about a concept. It is this extended structure that makes metaphor such a powerful and essential part of our thinking. Metaphors function as natural models, allowing us to take our knowledge of familiar, concrete objects and experiences and use it to give structure to more abstract concepts.

## Metaphor in the Interface

The characteristics of metaphor in our language are the same ones that govern how metaphor works in an interface. Just as metaphors invisibly permeate our everyday speech, so do they occur throughout the interfaces we use and design. Just as we use military terms to make the rather abstract process of arguing more tangible, so we use object and container terms to make the Macintosh file system more concrete. And just as we experience arguments as real conflicts, most Macintosh users believe that when they move a document icon from one folder to another, they are really moving the document itself (what is "really" happening is that a pointer to the file is being moved—of course, *pointer* is a metaphor too . . .).

Because we use metaphors as models, an interface metaphor that suggests

an incorrect model can cause difficulties for users. Consider the following scenario:

*A visitor arrives. As previously agreed, she phones from the lobby to tell me she has arrived. However, I've stepped away from my desk for a moment, so the voice mail system answers:*
*"I'm not here now. But if you'll leave a message after the beep, I'll get right back to you."*
*She does so, and is quite properly annoyed when I show up in the lobby half an hour later and ask what kept her.*

What neither of us knew was that there was a half hour delay between when she left the message in my mailbox and when it became available to me. Why the delay? The voice mail system resides on a machine in another building and, under conditions of heavy use, it can take as much as half an hour to notify me of a message.

It's easy to blame the system, but the real problem is that the system's metaphor has failed. The instruction manual and the on-line recordings inform users that they have *mailboxes* in which *messages* may be *left*. When the caller reaches the system, she hears a message in the callee's voice, saying "I'm not *here* now," followed by a beep—just as with an answering machine. A very clear model is presented: the caller has reached the callee's desk; the callee is not there; but the caller may leave a message in a *mailbox* that seems to act just like a conventional answering machine. Unfortunately, the model is incorrect. It provides no way for the users to understand that there might be a time lag between when a message is left in a mailbox and when the mailbox's owner can open it and find the message.

A more accurate metaphor would be an answering service metaphor. Messages would be left with an answering service, which would then forward the message to the appropriate person. Though this might not lead the user to expect a delay, it does provide grounds for appropriate suspicions when it becomes clear that something is wrong. The system need not even be redesigned for this metaphor to be used. Individual users could invoke the answering service metaphor just by having someone else record a message for them: "Tom's not at his desk. If you leave a message after the beep, I'll forward it to him." This is a small change, but it gives the user a different, more accurate model. No longer does the voice and language suggest that the message has reached its destination; instead, it's clear that an intermediary has the message and that it still must be forwarded.

The example of voice mail is a good one for several reasons. First, it clearly illustrates how an interface metaphor can provide the user with a model of the system, and how differences between the user's model and the real thing can cause problems. Second, it illustrates that a metaphor

can make a difference even when there are no graphics or text associated with the interface. Too many people think that interface metaphors exist, or are important, only when icons and graphics are used. Finally, the example is good because it's so universal. Nearly everyone who uses this voice mail system has a message that presents the wrong model.

Why don't people use the more accurate metaphor? There are several possibilities. Many users don't understand how voice mail really works—they believe that the system's metaphor is real, that messages are really left in the boxy phones sitting on their desks. When incomprehensible delays occur, well, that's just the way high-tech stuff is sometimes. But that's not the full story, because even those who understand how the system really works continue to present the wrong model. Some of these users may not understand the purpose of an interface metaphor. They blame the hardware, and it doesn't occur to them that something as nebulous as a metaphor could in any way compensate for something as real as slow hardware. Finally, even users who understand that an interface metaphor should provide the user with a good model of the system may not know how to go about finding better metaphors.

## Coming Up with Interface Metaphors

How do you come up with appropriate metaphors? This section describes the process of generating interface metaphors and applies it to a simple design problem. Although the design example is both hypothetical and oversimplified, it illustrates a useful approach to designing with metaphor.

Here's a brief overview of the process. Because the purpose of an interface metaphor is to provide users with a useful model of the system, the first step is to understand how the system really works. Second, because no metaphor can model all aspects of a system's functionality, the designer must identify what parts of it are most likely to give users difficulty. Finally, once the designer has identified the sort of model required, metaphors that support that model must be generated.

### Functional Definition

To create a model of something, obviously you have to understand how the thing itself works. This includes not only what the system can do but when that functionality is available to the user and how quickly the system can perform various functions. Some of this information may not be available in the system specification; nevertheless, it is essential, and the interface designer must either experiment with the system or probe the appropriate technical people to obtain it.

Now let's take a look at our example. Because we're using a fictitious, over-simplified example, the functional definition process is easy: we just make up the specifications. To make the example somewhat more realistic,

rather than starting from scratch, we will assume that we are going to add new functionality to the Macintosh.

In the Macintosh, data is shared between applications by copying and pasting. However, this can be cumbersome if the data in the source document is continually changing: every time it changes, you have to recopy it and then repaste it. Let's assume that our task is to automate this process. Users will be allowed to define links between parts of different documents, so that when a change is made to data in the source document, the change is copied automatically, sent over the link, and then pasted automatically into the receiving document. For the purposes of this example, we'll impose three constraints. First, links have directionality—that is, data can go only one way along a link: from source to recipient, or from beginning to end. Second, links can be one to many: one piece of data may be at the source end of many links, so that a change to it is sent to many different documents. Third, we'll assume that we can't guarantee that a change in data at the source end of a link can be transmitted instantly to the destination end of the link—there may be a time lag.

### Identify Users' Problems

The second step is to figure out what users have problems with. What aspects of the functionality are new to them? What may look familiar, but will really be different? The best way to do this is to observe users (see Gomoll's chapter in this volume). Watch them using similar functionality and see what problems they have. Describe what you're doing and see if they understand. Show them prototypes of your system and watch them try to use it (see Wagner's chapter in this volume for a description of prototyping). Each of these methods has its drawbacks, but any is better than just guessing.

What do users understand and not understand about the functionality of links? For the purposes of this example, we'll just guess. Users may not understand that links have directionality—that data can flow one way along a link, but not the other. Or, users may not understand that links are one-to-many: data at the source end of a link may show up in many different documents. There are other likely problems that would most appropriately be identified by working with users, but these two will serve our purposes.

### Metaphor Generation

The first step in generating metaphors is to note what metaphors are already implicit in the problem description. Because we use metaphors to talk about abstract concepts, it's almost certain that metaphors are lurking about in the description of the functionality. However, because the functionality is usually defined by technical people who aren't representative of end users, the metaphors are often inappropriate. Nevertheless, it is important to

identify these metaphors if only because unrecognized metaphors, may limit the variety of metaphors you generate.

In the description of our example, we are using a metaphor. We speak of the functionality in terms of *links between* documents. We say that links have *sources* and *destinations* and that data is *sent along* the links. It is important to recognize that there are other possible metaphors. Perhaps data could be conceived of as flowing through pipes. Or one could imagine electronic connections—a link could be created by wiring it up. Or a link could be thought of as a path, and special porters could carry data along it. But note that these metaphors—"pipes," "wires," and "paths"—are all special cases of the "links" metaphor. Each is a special type of *connection* metaphor. It may be useful to look for metaphors that focus not on connections, but on the properties of the data at either end of the link.

Mountford's chapter in this volume describes various techniques that can be used to generate new metaphors. However, one particularly useful approach is to focus on the user problems you've identified and look for real-world events, objects, or institutions that embody some of the characteristics that users find difficult to understand. These make good candidates for new interface metaphors.

For example, what are some real-world things that exhibit directionality? Rivers flow in one direction; TV images are transmitted from a broadcaster to a receiver; newspapers are mailed from a publisher to subscribers; forces like gravity and magnetism have direction. Similarly, TV broadcasts and newspaper editions also originate from one point and end up in many places at once; rivers generally do the opposite; and gravity and magnetism don't work at all.

It should be possible to generate dozens of potentially useful metaphors. For the purpose of keeping this example brief, we'll look at only three: links (because with the advent of commercial hypertext this is an increasingly popular metaphor), TV broadcasting, and newspaper publishing.

## Evaluating Interface Metaphors

Once several metaphors have been generated, it's time to evaluate and choose one through which to express the new functionality. For our purposes, we'll take our original metaphor—links—and the TV broadcasting and newspaper publishing metaphors described above. Here are five questions for evaluating the usefulness of an interface metaphor.

### Amount of Structure

How much structure does the metaphor provide? A metaphor without much structure may not be very useful.

One problem with the links metaphor is that it doesn't have much structure. If you ask users what a "link" is, you'll get a lot of different

answers—users don't have a clear notion of what a link is. Links can be one-way or two-way. Links may imply data flow, or they may just be physical connections. When users think of links, they may even think of chains, sausages, or golf. There's nothing in the links metaphor that suggests the directionality or one-to-manyness of the flow of data that the users need help in understanding.

In contrast, users know a lot about TV broadcasting and newspaper publishing. For example, newspapers have editions, subscribers, editors, delivery people, and delivery routes; they may be found at newsstands; and so on. TV broadcasting has networks, stations, channels, *TV Guide*, TV receivers, reruns, serials, shows, VCRs, and so on.

### Applicability of Structure

How much of the metaphor is actually relevant to the problem? What is particularly important here is not what is irrelevant, but things that might lead the user in the wrong direction or raise false expectations.

For example, in comparing TV broadcasting and newspaper publishing, note that the broadcasting metaphor may lead the user astray because broadcasting implies instantaneous transmission of the data; in contrast, everyone knows that newspapers take a while to get delivered.

### Representability

Is the interface metaphor easy to represent? Ideal interface metaphors have distinctive visual and auditory representations, as well as specific words associated with them.

This is another area where the link metaphor is weak. Even assuming that users understand that "link" means a "connection over which data can flow," what does a link look or sound like? In contrast, the broadcasting metaphor has many possibilities for representation; for example, a TV set for a receiver, a broadcasting tower for a transmitter, sounds for indicating transmission. The newspaper metaphor also has possibilities—one could imagine sounds and images of a printing press for the source end of the link and a newspaper as the receiving end. There are also rich vocabularies specifically associated with broadcasting (transmitting, receiving, tuning, reception range) and newspapers (publishing, issues, editions, circulation, delivery).

### Suitability to Audience

Will your audience understand the metaphor? A metaphor may satisfy all the other criteria, but if your users don't grasp the metaphor, it's useless.

Suitability would, for example, probably rule out the use of pointers (in the computer science sense) as a metaphor for links, even though they do indicate directionality. In evaluating suitability, it is once again important

for the designer to involve the user in the design process. It is exceedingly easy to become convinced of the suitability of a metaphor that, when tested on users, turns out to evoke completely inappropriate associations.

#### EXTENSIBILITY

What else do the proposed metaphors buy you? A metaphor may have additional bits of structure that may be useful later on.

For example, it would be nice if links worked across networks. Because both broadcasting and newspaper publishing work across large distances, it's likely that they have structures that can provide a degree of support for data sharing across machines. For example, one might imagine each server on a network having a *newsstand*, which would be the recipient of *publications* from other machines.

It's also important to notice ways in which the metaphor may extend itself. For example, the broadcasting metaphor might encourage a use of the link functionality in a way that is very different from the data sharing that it was originally envisioned as supporting. It's easy to imagine users deciding to use the link functionality presented via the broadcasting metaphor to browse among the different *channels* available, just as they would switch from one TV channel to another. This sort of behavior seems much less likely to occur with the newspaper publishing metaphor.

## What's Next?

Which metaphor should we choose for our example? It's not clear. To enable us to really address the question, the problem definition needs to be more detailed, more metaphors need to be generated and evaluated, and end users need to be consulted and observed throughout the design process.

Once a metaphor is settled on, it must be integrated into the interface. Although a discussion of this is beyond the scope of this chapter, one point should be made. Using the above criteria to select a good metaphor will be of little use unless the metaphor is used to its full extent. That is, having chosen a metaphor with a lot of structure, use as much of the structure as possible in the interface. Similarly, having chosen a representable metaphor, be sure to make full use of those representations.

Although the injunction to use and represent a metaphor fully may seem obvious, it's astonishingly easy to find examples where it has been ignored. For example, many HyperCard stacks start out using a book metaphor, usually by presenting a book-like background, but then fail to include such basic elements of books as page numbers, tables of contents, and indexes. Similarly, actions such as going to the next page are often represented by dissolving from one page to the next, rather than by using a visual effect to indicate that the page is being turned. These are all small details, but, as we saw in the voice mail interface, small details can make big differences.

## Summary

Although metaphors are everywhere, they are often difficult to notice. They are present whenever we speak or think about abstract concepts. Metaphors serve as natural models; they allow us to take our knowledge of familiar objects and events and use it to give structure to abstract, less well understood concepts.

Metaphors exhibit these same properties in interfaces. To the extent that an interface metaphor provides users with realistic expectations about what will happen, it enhances the utility of the system. To the extent it leads users astray, or simply leads them nowhere, it fails.

Designers need to do several things when working with metaphor. They need to notice what metaphors are already present in the system. They must understand the system's functionality. And, most important, they need to know which aspects of the functionality users may not understand. Armed with this knowledge, the designer can search for metaphors that best support the areas in that the user's understanding is the weakest.