

Oh What a Tangled Web We Weave: Metaphor and Mapping in Graphical Interfaces

William W. Gaver

Computer Related Design, Royal College of Art
Kensington Gore, London SW7 2EU, U.K.
gaver@rca-crd.demon.co.uk

ABSTRACT

The relations among graphical representations, computer functionality, and everyday objects are more complex than terms like “the desktop metaphor” may suggest. While metaphors in the everyday world highlight similarities between preexisting entities, interface metaphors create new ones. But new computer entities can also be created without using metaphor, when existing ones are combined via conceptual structuring and the results expressed via iconic perceptual mappings.. Naming such constructs involves yet another metaphor, however, between the functionality suggested graphically and that implied by the name. In sum, interface representations – which can only be called “metaphors” metaphorically – are complex and confusing, but this leads to a flexibility and power that may be lost if simpler mappings are used.

KEYWORDS: mapping, metaphor, semiotics

INTRODUCTION

Graphical interfaces are usually thought to be effective because they use metaphors to everyday objects in order to suggest their functionality. The problem with invoking metaphor as an explanatory principle, however, is that it oversimplifies the relations among computer functionality, graphical representations, and everyday objects. In this note I discuss some of the issues involved in creating representations of computer functions to show that the “desktop metaphor” is really a complex web of mappings.

METAPHORS IN THE EVERYDAY WORLD

Metaphors are linguistic devices in which the similarities between two things are highlighted by referring to them as

equivalent – for example, “I am an island,” or “painting the town,” or “family tree” (top of Figure 1).

Metaphors typically compare things that have preexisting identities, both conceptually and in appearance, and thus are easier to create linguistically than graphically. However, graphical metaphors can be created, for example by combining the appearances of the two concepts to be compared (e.g., the graphical “family tree” at the bottom of Figure 1). The ability to do this without changing the sense of the metaphor suggests that two different mappings are relevant. The metaphor itself is a *conceptual mapping* between the two ideas, while independent *perceptual mappings* relate each of the ideas to its representation as a word or picture (Gaver, 1988). Though in most cases metaphors create new concepts only temporarily, for the sake of emphasizing similarities among existing ones, occasionally new entities such as family trees are created that take on an existence of their own.

METAPHORS IN COMPUTATIONAL WORLDS

Metaphors work differently in computer interfaces than in the everyday world. Whereas metaphors usually highlight structural and conceptual similarities between existing entities, interface metaphors like “buttons” or “windows” serve to define new entities.

Two classes of attributes can be distinguished for these entities (Familiant & Detweiler, 1994): *appearance* (e.g., “round, appears to afford pressing”) and *functional* (e.g., “performs some task when activated”). In the everyday world, the perceptual mapping between appearance and functional attributes is often very close because objects’ physical structures are lawfully related to their perceptual appearance. For computer entities, in contrast, the perceptual mapping between their appearance and underlying functionality is almost entirely arbitrary and must be designed.

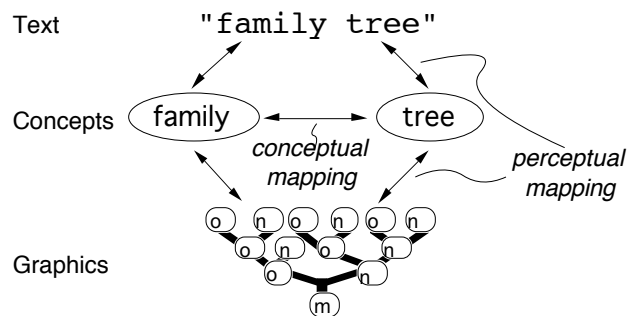


Figure 1. Metaphors are conceptual mappings independent from the perceptual mappings used to express them.

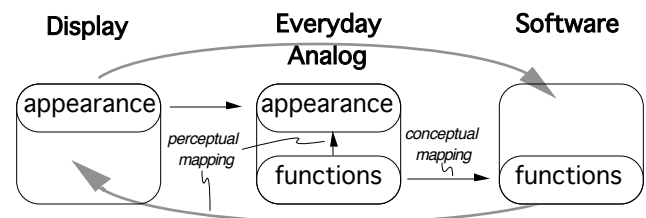


Figure 2: Metaphors fuse form and function.

After all, graphical displays (such as icons) have perceptual attributes but no functional ones. Computer entities have functional attributes but no perceptual ones. Interface metaphors allow the perceptual mapping to be created for an interface object via a conceptual mapping between the functional attributes of the computer and an everyday analog (Figure 2). By designing a display's perceptual attributes to be similar to those of an everyday analog, the display can simultaneously suggest a conceptual mapping between the functional attributes of the software and the everyday analog, and at the same time create a perceptual mapping between the display's perceptual attributes and the software's functional ones.

The result is that the metaphor acts to fuse the representation and software so that the graphics gains the functional attributes of the software, and the software the appearance attributes of the graphics. The two become one, producing a new sort of graphical entity that, unlike pictures, has functional attributes as well as perceptual ones, affordances as well as appearance.

Moreover, if this process of fusing graphics and software is successful, the role of the metaphor is complete. The computer entity no longer needs to be understood via an everyday analog, but instead can be understood directly as an item with functions and appearance. Thus to experienced users, interface objects such as buttons, windows, files and folders exist independently from their real world counterparts. After fusion, they are no longer metaphors.

BEYOND METAPHOR

Once new interface entities with both appearances and functions have been created using metaphor, they may themselves be joined via *conceptual structuring* to form more complex constructs. This is not a metaphorical process, but one of building from simple elements. An obvious example is the definition of data structures from lower-level variables. Similarly, interface objects having no analogs in the everyday world may be constructed from simpler elements. For instance, tool palettes are created by combining sliding surfaces, icons, and buttons to create new entities with no obvious analogs.

The perceptual mappings between the appearances of computer entities and those of their everyday analogs are seldom metaphorical. For instance, a conceptual button in the model world may be given perceptible form on the computer display by a picture of a button or by the word "button," neither of which involves a metaphor. Instead, a picture of a file involves an *iconic* mapping to the conceptual file, while the word "file" relies on a *symbolic* one (Gaver, 1988). True graphical metaphors (such as the family tree in Figure 1) are used only rarely for perceptual mappings, and when they are they usually take the form of *metonymy*, in which parts are used to stand for wholes (e.g., when the letter "A" is used to stand for a text-writing tool in graphics programs).

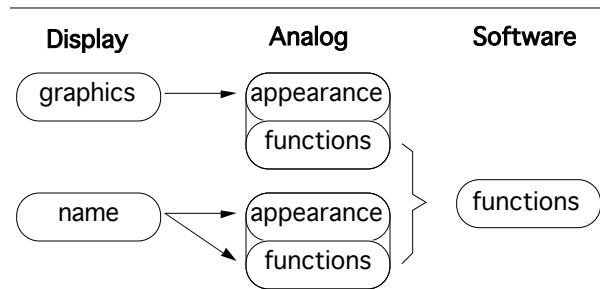


Figure 3. When names suggest different functionality than graphics express, software functions depend on the mix.

Finally, when iconic mappings indicate components of an interface structure, it is more appropriate to think about the *expression* of that structure than a metaphor. For instance, graphical indications that a button may be pressed or that palette items may be selected don't seem to rely on metaphor, but instead on the same sort of basic perceptual information that allows affordances to be perceived in the everyday world. At this level, the concept of metaphor applies only loosely and, perhaps, not very usefully.

MIXED METAPHORS: NAMES AND GRAPHICS

The situation becomes more complex still when graphical entities are *named*. In some cases (e.g., buttons) names and graphics suggest the same functionality. But in others (e.g., windows) the graphical devices used to create them – such as scrollbars, buttons, and the title bar – express different functions than those suggested by the name (Figure 3). Comparing these two sets of implied functions, however, allows the true functionality of software windows to emerge. For instance, scrollbars' graphical suggestion of changing something, and window's suggestion of views as something to be changed, are mutually responsible for expressing an aspect of graphical windows' functions.

From this perspective, names introduce a second layer of interface metaphor, one between the functionality expressed graphically and that suggested by the name. Metaphors from the everyday world may convey unwanted implications about functionality (e.g., buttons may not appear movable). Graphical expressions of novel structures (e.g., windows) may not be able to convey all the functionality being offered. Using both names and graphics allows two models to act as resources without either becoming overwhelming.

CONCLUSION

Calling the sorts of representations created by interface designers "metaphors" is itself something of a metaphor. As I hope to have illustrated, the mappings involved are more complex and varied than can be captured simply by that metaphor. Their complexity should be embraced, however, since it allows a flexibility and power that simpler, easier to conceptualize mappings may lose. In the end, we must resist the temptation to untangle our metaphorical web lest our interfaces become *too* simple.

ACKNOWLEDGEMENTS

I thank Martin Locker, conversations with whom inspired me to write this paper. I also thank Colin Burns, Gillian Crampton-Smith, and Anne Schlottmann for comments.

REFERENCES

Gaver, W. (1989). The SonicFinder: An interface that uses auditory icons. *Human-Computer Interaction* 4(1): 67-94

Familiant & Detweiler (1994). Iconic reference: Evolving perspectives and an organizing framework *International Journal of Human-Machine Studies*.