Computers that Roar

Computers, like other media, are metaphor machines: they both depend on and perpetuate metaphors. More remarkably, though, they—through their status as "universal machines"—have become metaphors for metaphor itself.

From files to desktops, windows to spreadsheets, metaphors dominate user interfaces. In the 1990s (and even today), textbooks of human–computer interface (HCI) design described metaphors as central to "user-friendly" interfaces. Metaphors make abstract computer tasks familiar, concrete, and easy to grasp, since through them we allegedly port already existing knowledge to new tasks (for instance, experience with documents to electronic word processing). Metaphors proliferate not only in interfaces, but also in computer architecture: from memory to buses, from gates to the concept of architecture itself. Metaphors similarly structure software: viruses, UNIX daemons, monitors, back orifice attacks (in which a remote computer controls the actions of one's computer), and so on. At the contested "origin" of modern computing lies an analogy turned metaphor: John von Neumann deliberately called the major components of modern (inhuman) computers "organs," after cybernetic understandings of the human nervous system. Drawing from the work of Alan Turing and Charles Babbage, Jon Agar has argued that the computer, understood as consisting of software and hardware, is a "government machine." Like the British Civil Service, it is a "general-purpose 'machine' governed by a code."[1]

The role of metaphor, however, is not simply one way. Like metaphor itself, it moves back and forth. Computers have become metaphors for the mind, for culture, for society, for the body, affecting the ways in which we experience and conceive of "real" space: from the programmed mind running on the hard-wired brain to reprogrammable culture versus hard-wired nature, from neuronal networks to genetic programs. Paul Edwards has shown how computers as metaphors and machines were crucial to the Cold War and to the rise of cognitive psychology, an insight developed further by David Golumbia in his analysis of computationalism. As cited earlier, Joseph Weizenbaum has argued that computers have become metaphors for all "effective procedures," that is, for anything that can be solved in a prescribed number of steps,

such as gene expression and clerical work.[2] Weizenbaum also notes that the power of computer as metaphor is itself based on "only the vaguest understanding of a difficult and complex scientific concept. . . . The public vaguely understands—but is nonetheless firmly convinced—that any effective procedure can, in principle, be carried out by a computer . . . it follows that a computer can at least imitate man, nature, and society in all their procedural aspects."[3] Crucially, this means that, at least in popular opinion, the computer is a machine that can imitate, and thus substitute for, all others based on its programming. This vaguest understanding—software as thing—is neither accidental to nor a contradiction of the computer as metaphor, but rather grounds its appeal.

Because computers are viewed as universal machines, they have become metaphors for metaphor itself: they embody a logic of substitution, a barely visible conceptual system that orders and disorders. Metaphor is drawn from the Greek terms *meta* (change) and *phor* (carrying): it is a transfer that transforms. Aristotle defines metaphor as consisting "in giving the thing a name that belongs to something else; the transference being either from genus to species, or from species to genus, or from species to species, or on grounds of analogy."[4] George Lakoff and Mark Johnson argue, "*The essence of metaphor is understanding and experiencing one kind of thing in terms of another*."[5] Metaphor is necessary "because so many of the concepts that are important to us are either abstract or not clearly delineated in our experience (the emotions, ideas, time, etc.), we need to get a grasp on them by means of other concepts that we understand in clearer terms (spatial orientations, objects, etc.)."[6] Lakoff and Johnson argue that we live by metaphors (such as "argument is war," "events are objects," and "happy is up"), that they serve as the basis for our thoughts and our actions.[7] Metaphors govern our actions because they are also "grounded in our constant interaction with our physical and cultural environments."[8] That is, the similarities that determine a metaphor are based on our interactions with various objects—it is therefore no accident that metaphors are thus prominent in "interactive" design. Crucially, metaphors do not simply conceptualize a preexisting reality; they also create reality.[9] Thus, they are not something we can "see beyond," but rather things necessary to seeing. Even to see beyond certain metaphors, they argue, we need others.[10] Metaphor is an "imaginative rationality": "Metaphor . . . unites reason and imagination. Reason, at the very least, involves categorization, entailment, and inference. Imagination, in one of its many aspects, involves seeing one kind of thing in terms of another kind of thing—what we have called metaphorical thought."[11] This imaginative seeing one kind of thing in terms of another thing also involves hiding: a metaphor, Thomas Keenan argues, means that "something . . . shows itself by hiding itself, by announcing itself as something else or in another form."[12]

Paul Ricoeur, focusing more on metaphor as a linguistic entity, similarly stresses the centrality and creative power of metaphor. To Ricoeur, metaphor grounds the possibility of logical thought. Ricoeur, drawing from Aristotle's definition, argues that change, movement, and transposition (and thus deviation, borrowing, and substitution) characterize metaphor.[13] By transposing an "alien" name, metaphor is a "categorical transgression . . . a kind of deviance that threatens classification itself."[14] Since metaphor, however, also "'conveys learning and knowledge through the medium of the genus,'" Ricoeur contends, "metaphor destroys an order only to invent a new one; and that the category-mistake is nothing but the complement of a logic of discovery."[15] It is a form of making, of poesis, that grounds all forms of classification.[16] This disordering that is also an ordering, a dismantling that is also a redescription, is also instructive and pleasurable—it offers us "the pleasure of understanding that follows surprise."[17] This movement from surprise to understanding is mirrored in metaphor itself, which is a mode of animation, of change—it makes things visible, alive, and actual by representing things in a state of activity.[18]

Computers, understood as universal machines, stand in for substitution itself. Allegedly making possible the transformation of anything into anything else via the medium of information, they are transference machines. They do not simply change X into Y, they also animate both terms. They create a new dynamic reality: the files they offer us are more alive; the text that appears on their screens invites manipulation, addition, animation. Rather than stable text on paper, computers offer information that is flexible, programmable, transmissible, and ever-changing. Even an image that appears stably on our screen is constantly refreshed and regenerated. Less obviously, computers—software in particular—also concretize Lakoff and Johnson's notion of metaphors as concepts that govern, that form consistent conceptual systems: software is an invisible program that governs, that makes possible certain actions. But if computers are metaphors for metaphors, they also (pleasurably) disorder, they animate the categorical archival system that grounds knowledge.

If theories of metaphor regularly assume that the vehicle (the image expressly used) makes the abstract tenor (the idea represented) concrete—that one makes something unfamiliar familiar through a known concrete vehicle—software as metaphor combines what we only vaguely understand with something equally vague. It is not simply, then, that one part of the metaphor is "hidden," but rather that both parts—tenor and vehicle—are invisibly visible. This does not mean, however, that software as metaphor fails. It is used regularly all the time because it succeeds as a way to describe an ambiguous relation between what is visible and invisible, for invisible laws as driving visible manifestations. Key to understanding the power of software—software as power—is its very ambiguous thingliness, for it grounds

software's attractiveness as a way to map—to understand and conceptualize—how power operates in a world marked by complexity and ambiguity, in a world filled with things we cannot fully understand, even though these things are marked by, and driven by, rules that should be understandable, that are based on understandability. Software is not only necessary for representation; it is also endemic of transformations in modes of "governing" that make governing both more personal and impersonal, that enable both empowerment and surveillance, and indeed make it difficult to distinguish between the two.