



Project
MUSE[®]

Today's Research. Tomorrow's Inspiration.

Interfacing by Iconic Metaphors

Marianne van den Boomen
University of Utrecht

Interface: What You See Is What You Get?

Operating a computer is quite easy nowadays. We all click our way through our desktops, we all know how to open our e-mail, how to save files in folders. Working with computers has become such a common practice that we barely realize how these actions are framed by metaphors. Our first association with “cleaning a desktop” has probably more to do with deleting unused icons and files than with polishing a work-table surface. Behind our personal computer we are all disciplined office workers.¹ Yet, while most operating metaphors are drawn from an office setting, we have no problem when they blend with metaphors from other settings, such as home, play, menu, or window.

Reading these metaphorical signs—no matter whether they come as words or pictures—and using them as tools usually goes on seamlessly: “what you see is what you get.” But, as we all know, sometimes things go wrong. Sometimes the sign/tools on our screen do not yield the expected results. Then the machine closes itself for us as users, and suddenly exposes its status as a black box—an opaque, unknown machinery between input and output.

In this article, I aim to track down what exactly happens when we are reading/operating a computer, and how metaphors enable or disable our access to black boxes. First, I introduce some basic

1. Victor Kaptelinin and Mary Czerwinski, eds., *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (Cambridge, Mass.: MIT Press, 2007).

notions about user interfaces and the way they organize and sometimes disorganize our access to our computer. Second, I undertake a close reading of computer icons as signs, and confront this with their functioning as tools. Third, I address icons as conceptual metaphors, and material metaphors. I conclude with an argument for a material semiotics, which is able to provide an account of the sign-tool-machine oscillation at work on our desktops.

A Computer Is No Coffee Machine

Any machine that can be operated by humans in variable modes comes with a so-called user interface: a specific arrangement of switches, buttons, and other operating tools. This mediating switchboard provides operational access to the machine, whether it be a dashboard, steering wheel, and pedals in the case of a car, the buttons for cappuccino or espresso on a coffee machine, or the keyboard and mouse by which we operate our PC.

All machines do their job by physically transforming energy and/or raw material into some product, effect, or event. While this also holds true for the computer, this machine does more. On top of its physical transformations (changes in the state of the electronic circuits inside), it stores *machine readable* representations of these states in computer memory, and translates parts of these into *human readable* symbols on the screen. This is what makes the computer special: it is both a physical-processing machine and a symbol-processing machine.

The exchange between the physical and the symbolical is accomplished by the common language of digitality, as the electronic circuit states and the human readable signs can both be translated into digital code.² Digital code embodies specific instructions (programs) for the machine, enabling it to load, read, and write entities in its memory. In order to enable humans to read specific parts of this memory, the computer needs a sophisticated device: a screen that shows both output and input in human readable form.

The screen displays visual representations of the inner state of our computer, ordered by pictorial icons, textual menus, and plane subscreens (windows)—the so-called graphical user interface (GUI). The displayed graphics may function as buttons to give commands to the machine. This is comparable with the pedals in a car, as with

2. Digitality is basically constituted by assigning numbers to discrete entities, thus enabling computational manipulation of these entities. Our contemporary computer systems work with these numbers in the form of binary digits (sets of zeros and ones) in order to compute, but computable digits need not necessarily be binary.

a desktop icon for starting your e-mail program or toolbar icons for commands such as “save” or “print.” They also may be feedback signs, indicators of how the machine is working—comparable with the speed display in a car, as with an taskbar icon indicating your are connected to the Internet. They also may, as a composite of signs, display a product—comparable to a delivered cappuccino by a coffee machine, as with a digital photo or a Word document. Such a composite data object is also stored in memory, but the output need not be an end-product as with the cappuccino. Since its digitality enables further manipulation, the output may function as input for a subsequent product by the use of commands such as edit, cut, and paste. These reiteration functions have no equivalent in a car or a coffee machine. Moreover, the output product might be in itself a tool for manipulating data products, as is the case with application programs, scripts, and plug-ins—also reiteration functions without equivalent in other machines. The computer is thus threefold: it is a physical- and symbol-processing device; it is a set of tools for the (re) production of data objects; and it is a tool-making machine, able to create tools that can be fed back into the machine.

The tools and products exist at several levels: as mutable data objects (files);³ as executable sets of commands (programs); and as interfacial signs (icons and buttons, menus and names, acoustic signals, moving bars). These levels are deeply intertwined and nested into one another: executable programs consist of sets of separate files, interfacial signs are representations of executable or executed commands in a program and are thus, in the last instance, also files. This complex machine of nested levels becomes even more complex when linked in a network with other computers, therefore also enabling transference of files, commands, and programs to other machines in the network.

To disentangle these complex layers, it is tempting to divide the screen representations into two kinds of entities, interfacial and content. After all, means and ends, tools and products, input and output are ontologically quite different, just as with the coffee machine: there is a clear difference between the button for cappuccino and the cappuccino itself. We never mistake the button for the coffee itself. But a computer is no coffee machine, and, in fact, it is quite common in computing praxis to take the button for the coffee itself.

3. The notion of “file” is, of course, a human-readable concept, a metaphor taken from the world of print. And even the concept of “data object” is already a metaphor, taken from the world of physical things.

The problem is that the interface tools are made from the very same material as the output products, and both are simultaneously displayed on the same plane: signs and symbols on the screen. We cannot read a text document without a word-processing program, and we have no access to our e-mail without the specific interface of an e-mail program. As Lev Manovich states in his seminal work, *The Language of New Media*: “[T]he choice of a particular interface is motivated by a work’s content to such degree that it can no longer be thought of as a separate level. Content and interface merge into one entity, and no longer can be taken apart.”⁴ He shows how these digital mergers of interface and content employ concepts taken from three media domains: print, cinema, and human–computer interaction (HCI). According to Manovich, this interfacial assemblage is not confined to the computer as such; it extends to what he calls a “cultural interface,” a general “human–computer–culture interface” by which we interact with culture at large.⁵

The Swallowing Screen

The screen is the plane where the three instances of print, cinema, and HCI come together, and this is probably the reason why Manovich considers the screen “the key element of the modern interface.”⁶ This is a substantial claim, and Manovich goes even further by narrowing down the screen to cinema: “Rather than being merely one cultural language among others, cinema is now becoming *the* cultural interface, a toolbox for all cultural communication, overtaking the printed word.”⁷

While this might hold true for visual culture at large, it certainly does not for a PC. Its interface not only consists of a screen, but also of a keyboard and mouse. These mediating devices even correspond neatly to Manovich’s triad of metaphorical domains: the keyboard interfaces print; the mouse, HCI; and the screen mediates graphic and cinematographic representations. The screen does indeed remediate cinematographic operations (framing, projecting, zooming, montage, and so on), but it also remediates print (in the form of files, letters, documents, fonts) and clickable HCI (cursor movements, selecting, clicking). The three interfacial devices are insepa-

4. Lev Manovich, *The Language of New Media* (Cambridge, Mass.: MIT Press, 2001), p. 67.

5. *Ibid.*, p. 70.

6. *Ibid.*, p. 68.

7. *Ibid.*, p. 86, emphasis in original.

rable and indispensable⁸ for PC operations, and there is no reason to privilege one above the other. The screen reassembles all interfacial processes, translating and returning them as visual entities on a flat visual plane. Hence the screen is just a graphical interface within a broader interface.

Manovich has his roots in film studies, and this might explain his inclination to overrate and reify the screen. However, the tendency to conceive the screen as *pars pro toto* for computers, contemporary media, and contemporary culture in general is not limited to film scholars.⁹ The screen appears to be a general, dominant visual metaphor¹⁰ for computing, in ordinary speech as well as in new media studies. This could be explained by the ability of the GUI to swallow up all other components in its visual representations, thus rendering irrelevant what remains invisible. What you see is what you get, and that's all there is to get. At least, that is what the GUI suggests.¹¹

Iconic Condensation

Hence the GUI is an interface within an interface, with a strong tendency to swallow up the other components. And it does not end here, since the GUI is in itself a nested compound, consisting of relatively autonomous smaller GUIs: operating-system interfaces, application-program interfaces, and helper-program interfaces—all with their own buttons, icons, and other operation displays. Since these tools are all visually represented, they are subject to the engulf-

8. Although we might dispense with the mouse when we learn how to use keyboard combinations to navigate the cursor. And when suffering from repetitive strain injury we could use a voice-recognition software interface in order to circumvent or at least minimize the use of mouse and keyboard. But dropping any interface for manipulating text and cursor would make the computer inoperable.

9. Cf. Sherry Turkle's metaphorical book title, *Life on the Screen: Identity in the Age of the Internet* (New York: Simon & Schuster, 1995). In Steven Johnson's *Interface Culture: How New Technology Transforms the Way We Create and Communicate* (San Francisco: Harper, 1997), the interface equals the screen with its visual metaphors. Even in Jay David Bolter and Richard Grusin's *Remediation: Understanding New Media* (Cambridge, Mass.: MIT Press, 1999), the basic principles of mediation—immediacy and hypermediacy—are both defined as "styles of visual representation" (p. 272).

10. More precisely, this is a metonymical trope, a synecdoche, as it takes a proximal part—the screen—to stand in for the whole, namely, a computer or "visual culture."

11. Contrary to a so-called command-line interface (MS-DOS, Unix, Linux), where the keyboard is the dominant interface device and metaphor/metonym. This interface is waiting for the user to type specific commands in order to let the machine do something. You don't see a visual approximation of what you will get; you see an empty space that has to be filled with your knowledge of commands. Here, what you *write* is what you get.

ing mechanism of any visual representation: in the very act of translating into visuals, they devour all other components and modes. By showing, they hide; by translating, they substitute; by representing, they reify.

This is especially the case with computer icons. These small pictures are, in fact, nothing but shortcuts to specific software commands in order to yield some result. However, in our daily computing praxis, this is rendered invisible wherever possible. Instead of referring to specific software commands, they cloak this, pretending that they refer to “places” on a computer (My Documents, My Network, Mailbox), to files, or to an application program.

Sometimes it is not even clear whether an icon refers to a place, a file, or a program. Take the mailbox icon, for instance. What does it represent for the ordinary user? “Well, my e-mail of course,” would be the typical response. But what exactly is implied in the conceptual shortcut, “my e-mail”? Does it refer to a specific program running? Or to a place on your computer, the mailbox, where your mail resides? Or to specific files, sent to you as e-mail messages? Or perhaps to all these notions combined? Usually, when your e-mail just works as you expect it should, these questions are irrelevant. Who cares? But sometimes these questions pop up, as in the following story, which shows how these different possible references may converge or diverge and confuse users.

Somewhere around 1997 a friend of mine had problems with her brand new cable connection. She asked me to have a look at her computer. When she clicked on the mail icon an empty inbox showed up.

“See, no mail. And I’m sure I have mail; I forwarded some mails from my office. That man who did the installation yesterday said everything was working—‘Just click and go, ma’am.’ But where is my mail?”

“Okay,” I said, “First, are you sure you are online?”

She pointed vaguely at the wires, still hanging chaotically all over the place, swirling to her desk. I hoped they were okay, yet, I meant the software connection. There was a network icon on her desktop, represented by a telephone connecting two computers.¹² And fortunately it appeared to be correctly configured; a double-click established the connection with her Internet

12. Over the years, the display of the connection icon has been changed. While it started with a plain telephone device, it evolved into the more appropriate telephone modem, and finally into an abstract device, standing for any kind of connection, enabled by a telephone modem, cable modem, or wireless network card. Average Internet users nowadays would perhaps not even understand the telephone as a network-connection icon, being not familiar anymore with dial-up accounts. A telephone icon nowadays usually stands for another application: voice over IP.

provider. And after clicking the “get mail” button in her mail program her mail streamed into her inbox.

“That’s it,” I said. “You first have to connect, with that icon.”

My friend was puzzled. “That telephone icon? I used that to dial up, but now I have a permanent connection, just like I have at work. There I always see my mail immediately, without having to click on anything. I don’t even have such an icon there.”

“Okay,” I said, “Apparently your computer at work is configured that way. That telephone icon stands for any kind of connection, be it by telephone line or cable. But you can bypass it. It is possible to connect automatically to your provider as soon as you start up your computer or your mail program. And also to fetch your mail as soon as you start your mail program. We can arrange that, if you want.”

And so we did.

That was all. Computer problems often cost hours of trial and error, but here we got a clear, comprehensible problem—just a small conceptual error, a small hidden step in between. Yet, this story is illustrative of how human–computer interaction works, how it often fails to work, and how this mostly has to do with conceptual misunderstandings, material configurations, and hidden steps in between.

The above anecdote might too easily be construed as a classic example of computer illiteracy, but that would miss the point. This is a story about literacy. My friend was acquainted with e-mail at work and at home, she knew the difference between dialing up and a direct connection, she was able to forward mail. In fact, she was *too* literate, especially regarding icon reading. Basically, she just took the icons literally, taking them for their referents: she took the mailbox icon for the mail itself, and the telephone icon for using the phone itself. For her, the mailbox icon was not referring to a process, a string of commands and programs set in action in order to obtain a computed result; instead, it functioned as a key to a specific place—her inbox—where she expected her mail to be, immediately. The icon was not read as reference, but as immediate access to the referent itself. The icon seemed to have swallowed up all references, transferences, and network labor involved—indeed, a shortcut.

It is important to note that this is not particular to my friend’s computer use alone, but is a general feature of iconicity. Computer icons function by iconic condensation: a condensation of reference, referent, and meaning into one visual sign, the icon. Hence the inclination to take the icon for a specific *state* (result, place, thing) instead of a referential button able to invoke a performative

*process*¹³ is very strong. In fact, it is part and parcel of the very function of icons, since desktop icons are shorthand for complex machine processes. This is the kernel of our contemporary graphical user interfaces: translating machine processes into human-readable stable signs, and translating user action into machine-readable processes. Icons can only carry out their signifying *and* executing job by concealing the involved complexities, that is, by representing stable entities.

In the case of my friend, the mail icon necessarily hides the complex nested processes it refers to: executing the mail program, including its configurations for a particular Internet connection, a particular mail account, and particular incoming and outgoing mail servers located at a particular Internet service provider. Instead, it represents a specific result of the process (received mail) located at a specific place (the mail box). The same holds true for the telephone icon: it conceals that it is referring to the execution of the Internet-connection program, including its configurations and settings; instead, it represents the contiguous telephone device.¹⁴

In short, both icons refer to machine processes to be executed, and not to a specific state, place, or thing. At the same time, they have to conceal this and represent stable, ontologized entities in order to provide a human-readable sign. This concealment goes further than just “non-representing”; it is an active process that involves specific labor against representation. I propose to call this “derepresentation.” We could then say that the icons on our desktops do their work by representing an ontologized entity, while derepresenting the procedural and material complexity. This is the way icons manage computer complexity, the task we have delegated to them. And this is why we are seduced, indeed compelled, to take icons literally, at “interface value.”

The process of *ontologizing* is constitutive for interfacial computer icons, and probably for any human-computer interaction. Some

13. The analytical distinction between *process* and *state* is elementary for programmers and software designers, but less so for semioticians, media scholars, and social scientists. Though structuralists distinguish between *diachronical* and *synchronical* analyses—the latter pertaining more to states and structures, the former more to processes of historical transformation of these states—this usually yields to different studies with different methodologies. The explicit terminological distinction of state and process *within* structures or transformations seems to be absent.

14. The telephone icon, in fact, enacts an already forgotten displacement. We learned not to take the telephone icon literally, since we “know” it stands for using the telephone modem and not the telephone device itself. In that sense, the telephone icon is not a metaphor (based on resemblance), but a metonymy (based on proximity).

scholars would call this “substantializing,”¹⁵ yet I consider substantializing (to treat as if a substance), reifying (to treat as if a thing), and anthropomorphizing (to treat as if a human being) as particular instances of ontologizing. Although ontologizing implies derepresentation, reduction, and fixation, it should be noted that it is a productive epistemological act of classifying, able to create objects of knowledge and intervention. In the act of ontologizing, dynamic properties (e.g., the prize of a commodity, behavior of people, output of a computer) are abbreviated, congealed, and represented as something stable (as thing, human being, state, or place). These ontologized entities need not necessarily emerge as isolated; they may imply or evoke their own external connections. When thus integrated in a broader discursive formation, this may result in the historical creation of new concepts along axes of knowledge, power, and ethics.¹⁶

Signs: Iconicity and Indexicality

I have argued that by the simultaneous enactment of representation and derepresentation, the computer icon tends to ontologize. It seems to implode as a sign, condensating reference, referent, and meaning. To further disentangle this ontologizing and condensating mechanism, we need to turn to the question, what kind of signs are computer icons? Would a theory of the sign be able to explain their “iconology”—that strange ontologizing capacity of icons? In order to flesh out these questions, I will take up C. S. Peirce’s theory of the sign. Peirce’s semiotics seems typically apt to solve the riddle of the icon, not only because it explicitly addresses iconicity, but also because his triadic sign analysis enables a connection to the world outside language.¹⁷

What I loosely called “reference, referent, and meaning” echoes faintly the three Peircian ingredients comprising a sign: namely,

15. See, for example, Werner Rammert, “Relations that Constitute Technology and Media that Make a Difference: Toward a Social Pragmatic Theory of Technicization,” *Society for Philosophy and Technology* 4:3 (1999).

16. Michel Foucault, *The Archeology of Knowledge* (London: Tavistock, 1972); Ian Hacking, *Historical Ontology* (Cambridge, Mass.: Harvard University Press, 2002).

17. Contrary to Saussure’s structuralist semiology, in which only two ingredients of the sign are distinguished: *signifier* (material embodiment of the sign) and *signified* (mental concept, produced by the differences in the chain of signifiers, arbitrarily connected to a signifier). In this linguistic conception of the sign, any notion of referring to something outside language is disabled. While this precludes the pitfalls of claiming a natural correspondence between words and things, it also excludes any conceptualization of possible relations to a world outside human language.

something articulating 1) a quality, enabling reference to 2) something else, and thus producing 3) meaning. As Peirce puts it: "A sign is something which stands for another thing to a mind."¹⁸ Peirce mostly used the terms *representamen* (perceptual material quality of the sign, the "signifier"), *object* (thing or phenomenon referred to, the "reference"), and *interpretant* (mental effect, thought, the "signified").

A sign, according to Peirce, can only function as a sign in the full dynamic trinity of representamen, object, and interpretant. These three determine one another in a recursive, nested manner; the third element (interpretant) always implies the other two, and the second element (object referred to) always implies a relation with the first (representamen). For example, the sign triad of a red traffic light would be made up by the representamen of "redness,"¹⁹ by the object of a busy crossroad, and by the interpretant of the thought, or meaning: "You are supposed to stop here in order to let the other traffic pass."

Firstness, Secondness, and Thirdness

That the Peircian sign comes threefold is no coincidence. The idea of a first, second, and third level is ubiquitous in Peirce's work, and he analyzed all kinds of phenomena (signs, logic, science) in terms of triads. This resulted in a general classification scheme of three basic "ontological" categories that he called "Firstness," "Secondness," and "Thirdness." Peirce was aware of his strange penchant for this numbered triad:

This sort of notion is as distasteful to me as to anybody; and for years, I endeavored to pooh-pooh and refute it; but it long ago conquered me completely. Disagreeable as it is to attribute such meaning to numbers, and to a triad above all, it is as true as it is disagreeable. The ideas of Firstness, Secondness, and Thirdness are simple enough. . . . Firstness is the mode of being of that which is such as it is, positively and without reference to anything else. Secondness is the mode of being of that which is such as it is, with respect to a second but regardless of any third. Thirdness is the mode of being of that which is such as it is, in bringing a second and third into relation to each other.²⁰

18. Charles Sanders Peirce, "Of Logic as a Study of Signs," in *Writings of Charles S. Peirce*, vol. 3, ed. Peirce Edition Project (Bloomington: Indiana University Press, 1986), p. 380. See also Robert Marty, "76 Definitions of The Sign by C. S. Peirce." <http://www.cspeirce.com/menu/library/rsources/76defs/76defs.htm>.

19. But also of "light" and the particular "thingness" of the traffic light. Signs may combine several instances of "Firstness."

20. Charles Sanders Peirce, "A Letter to Lady Welby," in *Collected Papers of Charles Peirce*, vol. 8, ed. A. W. Burks (Cambridge, Mass.: Belknap Press, 1958–66), p. 328.

Firstness is thus the level of qualities as such; Secondness, the level of relations; and Thirdness, the level of general laws integrating relations in a system.

The beauty of the scheme is that the categories can be reiterated at any level. Thus, while a sign consists of instances of Firstness (signifier), Secondness (signifier relating to an object), and Thirdness (signified), its Secondness—its possible ways of relating to an object—is in itself also threefold. This is Peirce's most well-known triad, based on the three basic relations a sign can have to its object: iconic, indexical, and symbolic. The *icon* (Firstness, of internal qualities) refers to its object by *resemblance* or *analogy*—for example, a portrait of a person, a traffic sign with a picture of a hollow in the road, or the map of a country. The *index* (Secondness, of external relations) refers to its object by an *existential* or *causal* connection—for example, smoke indicating fire, a fingerprint identifying a person, a sundial indicating time. The *symbol* (Thirdness, of general rules) refers to its object arbitrarily, by *habit* or *convention*—for example, the flag of a country, the logo of a company, or the alphabet for phonemes.

The triad can also be reiterated on the level of Thirdness: the interpretants can take on the mode of (First, quality) a *possibility*, (Second, relation) *fact*, or (Third, systematic law) *rule/reason*. Applied to the interpretants of traffic signs, the sign for a parking lot signifies a possibility, the sign for a hollow in the road signifies a fact, and the sign for one-way traffic signifies a rule. For all reiterated instances of First, Second, and Third, the principle of recursiveness holds: a Third always implies a Second and a First, and a Second always implies two Firsts. In fact, just as in calculus: $3 = 2 + 1$, $2 = 1 + 1$, $1 = 1$.²¹

Peirce's triadic recursive classification scheme provides a vocabulary to dissect and understand the working of signs at several intertwined levels, as it is scalable and dynamical. It gives an account of how a sign may be composed by other signs; how it may refer to a single object or a compound of objects, and how it may do so in different ways. It particularly accounts for nestedness: in every symbol (Third) there is also something indexical (Second) and something iconic (First). For example, a traffic sign is always a symbol, as it is taken up in a system of rules and conventions about forms and colors, but it may be indexically referring to a hollow in the road, and

21. And since 4 or more can always be decomposed in combinations of 3, 2, and 1, 3 is always enough, as Peirce firmly stated in "A Guess at the Riddle," in *Collected Papers of Charles Peirce*, vol. 8, ed. A. W. Burks (Cambridge, Mass.: Belknap Press, 1958–66), pp. 355–367.

it may do so by an iconic image resembling a hollow. Besides, signs may change—icons can become symbols; for example, the iconic picture of Che Guevara becomes a symbol when printed on a T-shirt, signifying not so much the person Guevara, but radical leftist sympathies. Likewise, symbols can be reiterated as an index; a yellow letter M is a symbol referring to the brand name McDonald's, but as a sign on a pole, it is indexical for the nearby McDonald's outlet. In short, the scheme accounts for semiotic transformations; interpretants may break loose and become new signifiers, invoking new objects and interpretants, new signs, with other relations and rules.

Indexical Symbols with Virtual Objects

Peirce's dynamic triad may be able to account for the interaction between state (Firstness), process (Secondness), and system (Thirdness) at work in computing semiosis, especially when it comes to computer icons. At first sight, these icons seem to be no Peircian icons at all, because they are more like symbols. After all, they are arbitrarily assigned; what a specific icon stands for—a file, a program—we have to learn, finding out by collateral experience. On another level, however, they appear as other kinds of signs. The icons display small graphics referring to their object, which can be done in three ways: based on resemblance (iconic), on existential relation (indexical), or on convention (symbolic). The mailbox icon would then be a Peircian icon, since its relation to its displayed object is based on a particular resemblance, a similarity between postal mail and e-mail. But the connection icon, which depicts a telephone device connected to a computer, would be more an indexical sign than an icon, since its representation is not based on resemblance, but on an existential relation. And the desktop icon for Microsoft's Internet Explorer, the branded stylized "e" (or the icon for Mozilla's Firefox browser, displaying a fox curled around a globe), would be a Peircian symbol, since the image is completely arbitrary and conventional. From one perspective then, all computer icons can be seen as symbolic signs, and from another perspective, they can be analyzed as icons, indices, or symbols. The latter concerns the object as represented by and in the sign; the former situates the object elsewhere, outside the sign.

In fact, these are two different kinds of objects. Peirce called the first one the *immediate object*—the object as represented by the sign—and the second one the *dynamical object*—the object in the "real"

world, that is, as interpreted instance. This object cannot be expressed by the sign, it can only be indicated, and it has to be learned.²²

This differentiation between objects enables us to flesh out how computer icons work. As we have seen, desktop icons materially refer to an act of executing machine code. From this perspective, all desktop icons are indexical signs: they refer to existential, physical chains of causation, to machine processes to be executed. Their dynamical object is thus code, but two kinds of code: indexically it refers to machine code, and symbolically it refers to human-readable code (e.g., “mail,” “file,” “program”). Hence computer icons are Peircian indices (referring to the dynamical object of machine code), wrapped in Peircian symbols (referring to the dynamical object of human code)—they are only contingently iconical (when representing the symbol by means of pictorial resemblance). However, in their signifying practice, they completely inverse this: they appear as primarily iconic. While computer icons are almost never genuine Peircian icons, they all exhibit what I have called “iconology”—reified iconicity. They do so by equating and substituting the sign with its immediate object of reference, thus negating its indexical reference to the dynamical object. In other words, they enact their iconicity by hiding their indexicality.

Again, we can see a Peircian triad shimmering through: the immediate object as internal Firstness, the dynamical object as relational Secondness. In order to understand the strange double dynamical object of computer icons (machine code and human code) I propose, in line with Peircian triads, a third kind of object: a future object, a not yet actualized object—an object to become, in short, a virtual object. This object is not represented in the sign, neither is it completely covered with the indexically executed machine code. The virtual object refers to what might be actualized by this code, referring to something that might be done with it by user action combined with machine action. It is not just your e-mail program, it is that you might have mail and can fetch it; it is not just your browser, it is the world wide web, opened up for you; it is where dynamical object, interpretant, and action are inextricably intermingled. It is true Peircian Thirdness: organized by systematic rules, produced by nested Firstness and Secondness, a possible configuration of actual relations in a complex system.

22. Peirce speaks of the “Dynamical Object, which, from the nature of things, the Sign cannot express, which it can only *indicate* and leave the interpreter to find out by *collateral experience*”; see C. S. Peirce, “A Letter to William James,” in *The Essential Peirce: Selected Philosophical Writings*, vol. 2, ed. Peirce Edition Project (Bloomington: Indiana University Press, 1998), p. 498.

Signs as Tools: Ready-to-Hand and Present-at-Hand

The proposed extension with a virtual object touches on a mechanism that Peirce's theory of the sign does not address explicitly. It concerns the moment that the sign becomes a *tool*: not something to be read and interpreted by a human being, but a thing in the hands of a user with which things can be done. Heidegger's distinction between tools *ready-to-hand* (*Zuhanden*) and tools *present-at-hand* (*Vorhanden*) can be of help here.²³ Heidegger points out how tools exist *Um-zu* ("in order to"); their being refers to a work to be done, in a system of other equipment and work. A tool we use is *ready-to-hand*, as it aligns with the activity in our hands and the goal in our minds. It exists seamlessly with its reference, its *Um-zu*, and does not call for further reflection—we do not think about a hammer when we need it, we just reach for it and use it. But when a tool does not function according to its *Um-zu*—for example, when it breaks down—it suddenly appears as a strange separate object. Then it is no longer *ready-to-hand*, but *present-at-hand*, open to questions and knowledge about what is the matter with it. We have to reconsider our engagement with the object, repair it, replace it, or change our goal altogether.

Desktop icons cannot break down as an object like a hammer might, but the modes of being as *ready-to-hand* or *present-at-hand* are still appropriate here. Icons are *ready-to-hand* when they yield the result we expect from them. We then routinely take the icons for granted; we neither think about them as tools nor as signs: they appear as *sign-and-tool*²⁴ together in their iconic *ready-to-handness*. The unconscious eye–hand coordination makes us even forget that they are connected to our hands and our clicking actions; the screen seems to be working on its own, by its own iconology. But when the icon does not yield the expected result, it is no longer *ready-to-hand*; it decomposes in a tool and a sign part, now both *present-at-hand*, raising questions about whether the tool–equipment chain is broken somewhere, or whether the sign was misinterpreted. In Peircian terms, this raises questions about its reference to the object, which seemed to be unproblematically iconic when *ready-to-hand*, but exposed its indexicality when *present-at-hand*.

From this perspective, the sudden turn of the desktop icon from an integrated *ready-to-hand* sign–tool into a decomposed *present-at-*

23. Martin Heidegger, *Sein und Zeit* (Tübingen: Max Niemeyer Verlag, 1979), pp. 66–83.

24. *Ibid.*, p. 79. Not be confused with Heidegger's notion of *Zeigzeug*, which pertains to signs as tools, *um zu zeigen* ("tools to refer").

hand object is not a disturbing inconvenience, but an opportunity—an opportunity to investigate taken-for-granted actions and notions, an opportunity to raise questions and learn something about the way digital equipment works, how it refers to other equipment and code, how it represents and ontologizes in iconicity, concealing its indexicality, how it can fool you, and how you can counter this. As Terry Winograd and Fernando Flores put it: “Breakdowns serve an extremely important cognitive function, revealing to us the nature of our practices and equipment, making them present-to-hand to us, perhaps for the first time. In this sense they function in a positive rather than negative way.”²⁵

Metaphor: Conceptual and Material

The vocabulary of ready-to-hand and present-at-hand enables us to view the computer icon as semiotically and ontologically taken up in an operational system, where a perpetual sign–tool–machine interaction is at work. Neither Peirce nor Heidegger, however, gave a full account of how signs may become tools or vice versa, let alone how the digital sign–tool condensation comes about. How can something be sign and tool in one? Signs are about referencing and signifying; tools are about doing and transforming. How can such an ontological gap be bridged? My claim is that this is achieved by metaphors.

Metaphors are special signs: they condensate two references by incorporating qualities from the one into the other. As such, they not just refer, but they also qualify; they qualify by transferring qualities from one domain to another.²⁶ This transference of qualities can be done by words, but equally so by other media forms (images, sounds) or even objects; they may stand for and qualify something else by blending their own qualities with the qualities of the signified. Hence metaphors are vehicles of transference, notably transference between conceptually and semantically different domains, but also between ontologically different domains—from words to objects or vice versa, from sound to images or vice versa, from digits to images or vice versa, from signs to tools or vice versa.

In order to track down these principles of transference, I will address below the so-called conceptual theory of metaphor to see what it can tell us about digital iconicity and transference. Subsequently,

25. Terry Winograd and Fernando Flores, eds., *Understanding Computers and Cognition* (Norwood, N.J.: Ablex Publishing, 1985), pp. 77–78.

26. The notion of qualification and objects-as-metaphor comes from Samuel Guttenplan, *Objects of Metaphor* (Oxford: Claridon Press, 2005).

I will turn to a theory of material metaphor in order to finally close the circle of sign–tool–machine oscillation.

Conceptual Metaphor

George Lakoff and Mark Johnson's theory of metaphor is based entirely on a conceptual transference paradigm.²⁷ Metaphor is conceived of as the transference of concept x taken from source-domain X and transported to target-domain Y , with the result that concept y is understood in terms of x .²⁸ Metaphors are defined as cross-domain mappings, sets of conceptual correspondences²⁹ across conceptual domains, which can be abbreviated in the formula: Y is X , or target domain is source domain.³⁰

We could apply this scheme on the mailbox icon. The underlying regulating conceptual metaphor is then "e-mail is postal mail," and the cross-domain mapping looks like this:

<i>Source domain: postal mail</i>	<i>Target domain: e-mail</i>
postbox	inbox of mail program
letters, packets	messages, attachments
sending and receiving	send or get mail button
sorting, disposing	distribution to folders, deleting
[postal distribution system]	[mail-server network at ISPs]
[delivery by postman]	[connecting to mail-server; fetch mail]

However, here we encounter some problems. The last two correspondences are indeed at work in the transference, but these are also exactly what is hidden in the iconic metaphor. They *are* and *are not* part of the conceptual mapping; they are what I have called "depre-sented," while the other correspondences are represented as such on the interface.

Note that this derepresentation does not coincide with Lakoff and Johnson's notion of downplayed or hidden parts of the domains.

27. Dubbed by George Lakoff as the "contemporary theory of metaphor," but also known as the "conceptual theory of metaphor," or the "cognitive theory of metaphor." See Lakoff, "The Contemporary Theory of Metaphor," in *Metaphor and Thought*, ed. Andrew Ortony (Cambridge: Cambridge University Press, 1993), pp. 202–251.

28. George Lakoff, and Mark Johnson, *Metaphors We Live By* (Chicago: University of Chicago Press, 1980), p. 5.

29. This correspondence is not confined to resemblance or similarity. While most other theories of metaphor hold that metaphor is a trope of resemblance, Lakoff and Johnson (*ibid.*, p. 151) claim that the perception of similarity can actually be *produced* by conceptual metaphor. Moreover, they conceive metonymy, synecdoche, and symbols as special instances of metaphorical concepts.

30. Lakoff, "Contemporary Theory" (above, n. 27), p. 207.

Hidden or unused parts in the metaphor “e-mail is postal mail” would be “stamps” in the source domain, and “viruses” in the target domain. These aspects remain unused in the conceptual transference, they have no cross-domain correspondence, and they are not necessary to establish the metaphorical concept of e-mail. Yet the aspects “postal distribution system” and “delivery by postman” in the source domain do have a correspondence in the target domain. Moreover, these correspondences are indispensable for a successful transference. But as such, they are not represented in the icon mediating between the source and target domains. Their operation is delegated to hidden processes, residing in a black box of software and machinery.

We may conclude that the contribution of Lakoff and Johnson’s theory to the understanding of computer icons remains at the level of the graphical user interface. It can provide an account of how the mailbox icon connects and condensates the source domain of postal mail with the target domain of e-mail, but the model has no room for the material, indexical operations involved in the transference: a connection with an Internet service provider, a mail client consulting a mail server, and the subsequent transport of mail to the user’s PC. In schematic form, it shows an input and output mechanism represented on the screen interface, and a loop mediated by the metaphorical mailbox icon, which deprents the indexical references in the black box (Fig. 1).

Usually, when everything works smoothly, this is no problem, the sign-tools are so ready-to-hand that we can afford to forget them as tools and equipment and read them as signs. The user can take them at “interface value.” However, reading and understanding the icon is actually not enough to let it do its job; while this suffices for linguistic or visual metaphors, it will not do for the desktop icon. The desktop icon needs *action* to complete its signifying task successfully—not only tool-action by the user, who must at least click on the icon, but especially action inside the machine and the network. In that sense, the icon is a two-faced Janus: one side is faced toward the user, who must be able to read, understand, and operate the icon; and the other is faced toward software and machine processes. But the user is only able to read the icon as far as the other side is concealed and deprented—black boxed. And this is exactly what the icon does by its icontology: equating the sign with its immediate object, effacing its indexical relations with its dynamical and virtual object.

Lakoff and Johnson’s theory of metaphor explains this partially by focusing on the cognitive understanding and reading of the icon,

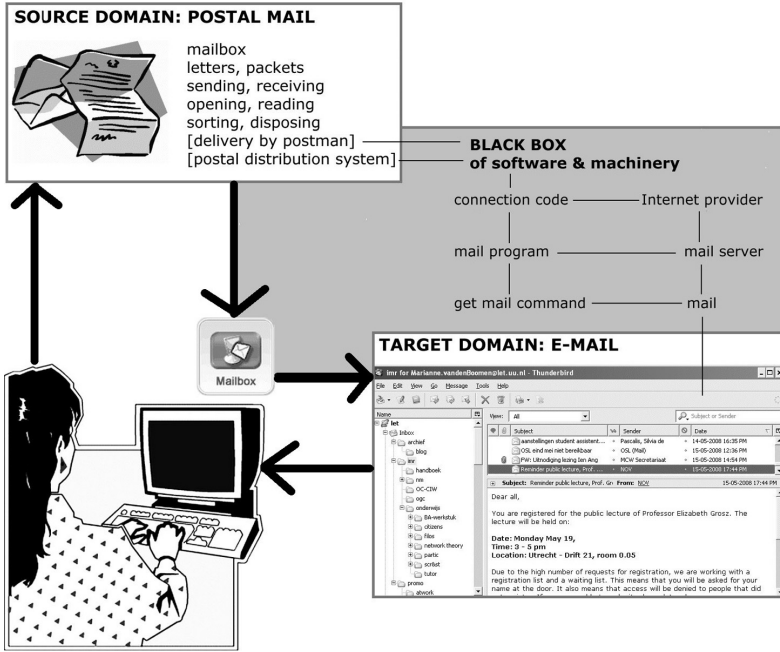


Figure 1. Input-output mechanism of the conceptual metaphor “e-mail is postal mail.”

on its relation with the immediate object; it gives an account of the metaphorical condensation of source and target, of the blending of reading and meaning. However, this only holds for sign-tools ready-to-hand, which render their output seamlessly; when a sign-tool breaks down and becomes present-at-hand—a quite common mode of being in computing praxis—we can no longer ignore its material indexicality beneath, its relation with its dynamical and virtual object. Then we have to open up the black box and check where, in the deprented chain of equipment, a link is broken. In short, Lakoff and Johnson’s theory needs to be extended in order to provide a full account of metaphors mediating between the human-readable and the machine-readable. What we need is a more material account of the iconic metaphor.

Material Metaphors

The problem of the computer icon as a sign-tool and its oscillation between its tool-being and its sign-being arises from its multiple materiality. It has, as do all signs, a materiality as sign, in this case

in the form of colored pixels on a screen—"flickering signifiers"³¹ as Katherine Hayles calls them—but it also has a materiality as a tool that is able to invoke material transformations inside the machine and the network. It is this latter materiality that is actively suppressed and derepresented in conceptual metaphors.

Hayles proposed the concept of "material metaphor" in order to include this notion of materiality.³² She defined material metaphor as the instance of metaphor where the transference does not take place between words or concepts, but between signs and material apparatus. With this notion, we may be able to open up the black box in the sign–tool–machine network invoked by computer icons, since it focuses on transference from sign to physical artifact—a transference that implies more than just the transference of conceptual meaning or qualification.

The materiality of a material metaphor does not reside in the materiality of the sign itself—after all, all signs are material, and metaphors may take any material form provided by any medium: linguistic, visual, acoustic, physical objects,³³ and so on. The materiality of a material metaphor resides somewhere else: in the change in a state of affairs in the material world, invoked by a material "execution" of the metaphor. The material metaphor has this possibility of material execution built in, along with its conceptual signifying power.

The material execution is nowadays usually enacted by computers, but Hayles notes that the occurrence of material metaphor is not confined to digital machines:

the transfer takes place not between one word to another but rather between a symbol (more properly, a network of symbols) and material apparatus. This kind of traffic, as old as the human species, is becoming increasingly important as the symbol-processing machines we call computers are hooked into networks in which they are seamlessly integrated with apparatus that can actually do things in the world, from the sensors and actuators of mobile robots to the semiotic-material machinery that changes the numbers in bank accounts.³⁴

31. N. Katherine Hayles, "Virtual Bodies and Flickering Signifiers," *October* 66 (1993): 69–91.

32. N. Katherine Hayles, *Writing Machines* (Cambridge, Mass.: MIT Press, 2002).

33. Physical objects as metaphors qualifying something else are, for example, textile swatches for a suit, or a weaving tree in a storm for a state of mind; see Guttenplan, *Objects of Metaphor* (above, n. 26).

34. Hayles, *Writing Machines*, (above, n. 32), p. 22.

The numbers in bank accounts are good examples of material metaphors that are not necessarily electronically mediated (though, of course, always “computed,” in the sense of “calculated”). These numbers do not just refer *conceptually* to your money resources, but also *materially*—change the symbols and you really do get materially richer or poorer.³⁵ Numbers as such are not material, but their banked transference effects surely are.

Still, we could think of material metaphors that are also crudely material as a sign, in the sense of being a physical object. Speed bumps on the road, for instance, are material objects signifying that you have to slow down, and they enforce this when “executed” by traversing them. Visible surveillance cameras are also such “double” material metaphors, as they are material objects saying, “Watch out, you will be watched,” and they do so indeed when invoked. A material metaphor can take on any modality—textual, visual, acoustic, gestural, physical—and can nestle itself in any medium or device as a semiotic–operational connection between a sign and a material event.

Hence a material metaphor does not just signify, it is also able to evoke acts in the material world. In other words, it condenses two references, but in another way than plainly linguistic metaphors. The difference is that material metaphor blends a conceptual and a material reference, not two conceptual references. It exploits the condensation of iconicity and indexicality in one sign–tool. Its instantiation often takes the form of a switch or button, as with desktop icons, which are visual signs and operational buttons in one. We might also think of other symbolic–operational signs such as hyperlinks on a web page (conceptually referring and materially transferring files in one) or a fire alarm that automatically connects to the fire department (indicating fire and evoking its suppression).

In fact, any button or switch can become a material metaphor when its signified includes, yet goes beyond, its instrumental tool-being. Material metaphors thus create two kinds of surpluses, depending on the point of view: a surplus of qualification, on top of its instrumental indexical function; and a surplus of material-physical action, on top of its symbolic representation. In their two-way direction, material metaphors function as switches between symbolic-semiotic systems (sign systems, languages) and material systems

35. Specific numbers (as on bank accounts, but also as school grades, telephone numbers, or IP numbers) can also be conceived as Peircian signs, connecting their numeric representamen to specific dynamical objects (respectively, money resources, school performance, telephone device, hardware device).

(the Internet, banking systems, telecommunication networks, surveillance circuits, but also nondigital networks such as equipment, electricity, and transportation infrastructures).

The ongoing proliferation of intermediations, induced by the combination of electricity and digitality, results in more and more material-semiotic, techno-social blends such as data-based friendship networks, online communities, virtual companies, and other dispersed though networked constructions. This can all be seen as evoked by the productive labor of material metaphors. In this situation, where material tools become more and more sign systems, and sign systems become more and more tools, we are not just transferring messages anymore, neither are we just doing things: we are doing things in the world and with the world by transferring messages, thus extending the sign-tool machinery.

Hacking the Black Box

Compared to the notion of conceptual metaphor, which mediates between conceptual source and target-domains, the material metaphor focuses on other kinds of transference. It often presupposes and parasitizes on the transference between conceptual source and target, but most important is its transfer from material force into event. While the conceptual metaphor of “e-mail is postal mail” can explain what happens inside the head of the sign-interpreter, the material metaphor leads us via the hands of the tool-user through and behind the interface, inside the black box of digital machinery, inside the material domain of referencing to dynamical and virtual objects.

This domain consists of software at different levels (client programs, server programs, computer languages, network protocols, commands, data packets), but also of hardware (PCs, ISP-servers, cables, modems, routers, switches). It is a network of equipment in which all devices refer and transfer to one another. This machinery is not just a neutral, enabling means for executing the software; it thoroughly frames our way of thinking and doing things, in aligning our heads and hands with data objects and virtual objects.

We humans are practically blind in this domain of digital machinery. We have delegated the relevant intentional acts to code, which we can execute but not grasp directly. We are simply not equipped with a perceptual or cognitive apparatus to read the digital, to read patterns of numbers and infer their meaning. However, we do have mediated access to the digital, because machinic calculations and translations can be ontologized and represented to us as readable signs—that is, as material metaphors. We are able to read

and act upon parts of the code, as far as it is represented as a sign-tool. This may be done by icons, but also by textual configuration menus (“mail-server settings”) or other metaphorized commands (“get mail”; “send mail”). And if we are suitably trained in computer languages, we are even able to read and modify the program itself—not just its operation or configuration, but its source code.³⁶ At a more mundane level, we can have basic notions of software objects without needing coding skills: we can loosely bear in mind conceptual frames like “connection program,” “mail program,” “service provider,” and “mail server.”

These kinds of representations and translations are conducted by material metaphors at several levels: hardware design (slots, buttons, ports, drives), interface design (icons, menus, pointers, arrows, symbols, file names), software modalities (assemblers, compilers, source code, scripts, executable applications, files, tables in databases), and protocols (sets of rules for translation and transfer). While these instantiations of material metaphors mostly do their work unnoticed and ready-to-hand, they are, however, near at hand. In fact, they are the keys of the black box, not only able to close and conceal, but also to open and reveal. They can be used for what could be called “epistemological reverse engineering”: tracking down the subsequent translations and transferences—this connects to that, this translates that into something else, then it is transported to that, and so on. In other words, material metaphors not only transfer meanings and acts and not only perform the necessary indexical labor of translations among energy, hardware, code, and events, they also provide analytical apertures in the digital black box, thereby allowing us to peek inside.

As we have seen, the iconology of the graphical user interface produces closed black boxes by translating the machinery into nested signs, translating what you get into what you see, thereby representing the indexical transferences at stake. The ready-to-hand tool appears as a “ready-to-eye” sign, concealing its tool-being. As long as we conceive these sign-tools as plain iconic signs or conceptual metaphors, we get fooled by their iconology. What we need is a shift from sign to sign-tool, from iconicity to material indexicality—that is, a shift to material metaphor. An analysis in terms of material metaphor not only accounts for the sign part, but also for the tool part, and most of all for the sign-tool oscillation at work in the digital machinery. To employ a last conceptual metaphor here,

36. That is, as far as we are allowed to; for ordinary users, this is only the case with so-called open-source software.

a material metaphor is an epistemological hacking tool, enabling us to hack the black boxes of metaphor, code, and machinery.

Acknowledgments

With special thanks to Charles Forceville, Anneke Smelik, Piet Zweers, and the members of the Metaphor Club for their critical and inspiring comments. A shorter version of this essay has been published as “Interfacing by Material Metaphors: How Your Mailbox May Fool You,” in *Digital Material: Tracing New Media in Everyday Life and Technology*, ed. Marianne van den Boomen et al. (Amsterdam: Amsterdam University Press, 2009), pp. 253–266.