The greatest thing by far is to be a master of metaphor. It is the one thing that cannot be learnt from others; and it is also a sign of genius, since a good metaphor implies an intuitive perception of the similarity in dissimilarities.

--Aristotle, ca. 330 B.C. [2, 1458b]

The early Fathers could not properly represent some things by the images of others unless trained, as I have said, in the hidden alliances and affinities of all nature.

--Pico della Mirandola, 1489 [6, p. 79]

For in many ways, the modern theory of computation is the long-awaited science of the relations between parts and wholes. . . . Computer science has such intimate relations with so many other subjects that it is hard to see it as a thing in itself.

--Marvin Minsky, 1979 [4, p. 393]

An administrator at my institution was recently advised to talk with the head of Data Processing. "I can't; she doesn't speak English," replied the administrator, to a chorus of nods and grins from eavesdropping colleagues.

English is in fact the data processing supervisor's native language, but she is by no means the first computer professional to be accused of incomprehensibility. Students complain that computer instructors cannot communicate; columnists call the language of computer scientists "computer illiteracy"; satirists represent "CompuSpeak" with invented monstrosities such as "user-friendly liveware" (salesperson) and "anthroperipheral interface error" (human error). One introductory textbook compares computer terminology with the Latin of medieval scholars—an impractical, elitist tongue unintelligible to ordinary people.<<1>>

In self-defense, computer professionals may point out that any technical field needs technical terminology and that their own jargon is unusually user-friendly [7]. The defense is valid, but it fails to convince our accusers—and for good reason. The technical terminology of computing differs from that of other fields in a way that makes it more difficult even while appearing simpler. Computing has given us a new and paradoxical kind of jargon, based on an apparently unscientific, non-technical principle: metaphor.

Anatomy of a glossary

Like languages in general, the "sublanguages" of special areas can be created in various ways. First, new terms can be introduced, usually from foreign roots, affixes, or proper names. For instance, plant biologists appropriated "Aufwuchs," a German term for "luxuriant growth," to designate organisms that coat submerged materials; ecologists combined two Greek affixes to produce "xerophile," or drought-loving; physicists named a measure of engine efficiency the "Carnot cycle" after its formulator. Second, existing but uncommon words may be appropriated and perhaps combined in a new phrase, such as "inert gas" (chemistry), "differential equation" (mathematics), "revolution of rising expectations" (political science),

or "theater of the absurd" (literature). Where the first method produces obviously specialized language, inscrutable for the outsider, the second yields terms that are somewhat more familiar but still in need of explanation. A third method is to use words familiar from other contexts but to give them new meanings, as the psychoanalyst did with "working through," the physicist with "charmed," and the geologist with "fault." A term formed in this third way needs explanation, like the products of the second method, but unlike them, it does not seem in itself to be part of a technical vocabulary.

The three methods result in three varieties of jargon, which may be called neologisms, technical terms, and redefinitions. Table 1 summarizes the categories and provides examples from computer science. All academic fields probably use all three categories, but not in the same proportions. We associate exotic neologisms particularly with the natural sciences, for instance, and mouthfilling "technical terms" with the social sciences. What of computer science? Does it share with physics a fondness for neologisms, or does it rely like economics primarily on abstract words and phrases?

To answer that question, I compared the vocabularies of six introductory textbooks: two from computer science (a programming text and an introduction to information processing), two from the natural sciences (ecology and physics), and two from the social sciences (economics and psychology). From the glossaries of those books I chose terms at an arbitrary fixed interval; I then assigned the terms to the categories in Table 1.

Table 2 summarizes the results, using the three general categories described in the previous paragraph. As expected, the natural science textbooks gloss mostly neologisms, and the social science books mostly technical abstractions. In contrast, the programming and information-processing texts rely heavily on "redefinitions," employing a far higher proportion of such terms than do any of the other books.

Second-hand jargon

Redefined terms possess a split personality. On the one hand, the terms themselves are familiar. Of the forty redefined terms in the sample from the programming glossary, as listed in the Appendix, at least thirty-five have long been in general use: argument, block, comment, constant, device, directive, driver, evaluate, external, field, and so forth. Hearing them, the outsider may feel at home.

On the other hand, within computer science these long-familiar terms have taken on new identities, which usually cannot be deduced from the old ones. The common meanings of "argument," for example, do not reveal what the term means to a programmer: specific values used in generic commands. Someone who knows a "block" only in its ordinary sense, as a visible unit or segment, cannot identify a block of records or of program statements. "Be sure to comment your code adequately," makes little sense unless one knows the relationship of program comments to source code, compilers, and executable statements. Then again, computer science makes peculiar use of source, code, compile, executable, and statement. In short, the newcomer is likely to feel not so much that computer scientists speak a different language as that they have rewritten the dictionary.

Among students of language, the application of "a word which in ordinary usage signifies one kind of thing, quality, or action . . . to another, without express indication of a relation between them" [1], has long been known as the use of metaphor. Literary scholar W. B. Stanford describes metaphor as follows: "A term (X) normally signifying an object or concept (A) [is used] in such a context that it must refer to another object or concept (B) which is distinct enough in characteristics from A to ensure that in the composite idea formed by the syntheses of the concepts A and B and now symbolized in the word X, the factors A and B retain their conceptual independence even while they merge in the unity symbolized by X" [8, p. 101].

Stanford's elaborate formulation can be applied precisely to the redefined terms of computer science. For instance, block (Stanford's "word X"), normally signifying a spatial unit or group of objects ("concept A"), is applied to a collection of program statements related both physically and syntactically ("concept B"). Run, which traditionally denotes rapid and unrestricted physical action, comes to mean also the automatic and--one hopes--unimpeded execution of commands.

It must be admitted that the two meanings of a computing term may not "retain their conceptual independence" for all speakers. That is, the computer professional uses block, connector, field, window, and document without remembering what those terms meant B.C. (Before Computers). Concept B has swallowed Concept A: the metaphors are dead. Meanwhile, they have not yet come to life for the novice, who waits in vain for Concept B to emerge from the conceptual slumber. But for anyone who understands their full range of meaning, such terms are, in origin and structure, metaphors.
The inevitability of metaphor

That computer scientists should model their vocabulary on that of poetry may indicate that they possess overactive imaginations—a suggestion that some of their colleagues would readily confirm. Alternatively, their penchant for metaphor may reveal their ignorance of the foreign terms that could generate neologisms. More charitably, we could infer that they use metaphors because they wish to lead nonspecialists into their territory gently, by way of familiar concepts.

On the other hand, Richard Kittredge, a specialist in technical sublanguages, argues that the characteristics of a sublanguage reflect not simply the idiosyncrasies of its users but the structure of knowledge in the field itself [3]. If he is right, metaphor may be in some way particularly appropriate to computer science.

A review of the definition of metaphor bears out Kittredge's generalization. Metaphors, we are told, partially identify concepts that are normally regarded as distinct. We might expect therefore that they will arise when links are being forged or perceived between diverse objects or ideas. For instance, a sublanguage will be metaphoric if its speakers habitually find parallels between their field and others.

And indeed, computer scientists spend much of their time seeking and extending such parallels. Their work consists largely of studying, creating, and controlling electronic systems that imitate other systems: logic diagrams, people doing arithmetic, chess-players, children playing with blocks, ledgers and files, the flow of traffic through an intersection, the behavior of a nation's economy over a certain period of time. That is,
computer scientists build simulations and models.

Historians tell us that the philosophers and poets of the Renaissance used metaphor to expound detailed correspondences among the universe, the "body politic," and the human microcosm [9]. In the words of one Renaissance philosopher, "Versed in all things and inspired by that Spirit which not only knows all these things but made them, they aptly symbolized the natures of one world by those which they knew corresponded to them in the other worlds" [6, p. 79].

So too, perhaps, the creators of computational microworlds use metaphor because they see reality through the lens of analogy.

Indeed, computer scientists can be expected to cling even more tenaciously to metaphor than did the analogy-makers of the Renaissance, for their microworlds are inherently symbolic. When a Shakespearean character calls the Roman Senate the belly of the State or brands a politician "a disease that must be cut away," he identifies the human body with the nation, but in such a way that they can still be clearly distinguished. Separate terms for Concepts A and B are still available. But the objects and processes of computing can scarcely be named apart from the elementary objects and processes that they model. Granted, we can call records "identical sequences of variously structured memory locations," or windows "buffered and overlapping screen displays." But such circumlocutions are not only verbose but imprecise. The commands that set up the electronic data-structure called a "record" were intended to reproduce the structure of office records. Similarly, the meaning of a screen-display window lies less in techniques than in the effect the techniques produce—the opening of a window into another realm.

Thus the computer is a symbol-manipulator, as Joseph Weizenbaum says, not only because it can represent virtually anything, but because it is designed to represent other things—to represent, in fact, many of the fundamental processes and relations of physical and mental reality [11, p. 74]. According to the literary critic A. D. Nuttall, "There appear to be certain areas of discourse where we can never afford to give up the metaphors we have inherited. Perhaps the principal examples of this sort of area are, first, theology, and, second, language about the mind. . . . Your metaphysician is a great metaphorist" [5, p. 22]. So, I would add, is your computer scientist. The vocabulary of computer science is incurably metaphoric because its subject matter is paradigmatic.

Living with metaphors

Whatever their metaphysical pedigree, computer metaphors are surprisingly troublesome in everyday communication. For the layperson they can be semantic wolves in sheep's clothing, more dangerous than terminology that openly announces its strangeness. Instructors are less likely to define "field" or "access" than "auf-wuchs" or "revolution of rising expectations," and students are less likely to demand explanations. Moreover, when a familiar-sounding term is defined, students may not grasp the definition, for past associations obstruct new ones. Long familiar with "drive" and "argue," the programming student may not even hear the instructor's explanation of device drivers or the arguments of functions.

The split between familiar and technical meanings has its comic side. As cartoonists know, it generates puns: one monk points to another who is busy with a microcomputer and says, "This is our chip-monk." But for the would-be computer user, ambiguous jargon is less funny. Recently I overheard a young man explaining to a middle-aged librarian how to remove the disk from a Macintosh. "Well, first you go to where it says 'file,'" he began, only to be interrupted with, "Now, in what sense are you using the word 'go'?" The question might have been amusing if the librarian's hands and voice had not been trembling with anxiety.

Both parties to such a miscommunication are likely to believe that they are using words in their proper or literal senses. For the computer professional, cursor movement is indeed movement, and data fields are areas; thus the uses of "go" and "field" in their new context are transparently appropriate. The new user will retort that "go" means "to move physically" and "field" literally means "an area of ground or study." That is the way of metaphors: depending on one's frame of reference, a metaphor's "literal" meaning may be either its older meaning or its extended one.

We might conclude that computer science should renounce metaphor altogether in favor of more direct terminology. But it is difficult to imagine a concise, nonmetaphoric alternative to "go" or "field." I have argued in fact that metaphors are indigenous to computer science.

If we cannot banish computer metaphors, we can nonetheless try to tame

<<2>>Pico is referring to the early patristic writers, but the practices he describes were the goal also of his contemporaries.
them—to render them less confusing. To begin with, we can acknowledge that they are indeed metaphors. In everyday language, unrecognized metaphors pose few problems because they are unrecognized universally. No one now remembers—or cares—that "interval" once meant "the space between two pillars," or that "daisy" comes from "day's eye," but in a new and changing area such as computer science, metaphors that some speakers no longer recognize may be all too fresh for others. The professional who treats "go" as flatly equivalent to "move the cursor" and the novice for whom "go" simply means "move one's body" are stranded on either side of the difference between "concept A" and "concept B." Novices cannot bridge the gap, for they can neither forget Concept A nor pass by an act of will to Concept B. It is computer professionals who must build the bridges, and we can do so only if we can see the distance to be bridged. That is, we must remember and acknowledge the original meanings of our metaphoric terms.

In addition, we can make our metaphors' new meanings concrete. Although computer metaphors refer ultimately to abstract categories of space and thought, most of them also designate specific procedures or objects in computing. It is those procedures and objects—the metaphors' new context—that the novice finds baffling.

To elucidate the context, we can use several techniques. One, which many instructors use already, is to display examples: sample records and fields, typical arguments, and so forth. Another is to make a term operational, associating it with particular actions: the keystrokes or mouse-movements that correspond to "go," the words or punctuation that delimit a "block." Finally, we can map a term's new context by mentioning its opposites. We can point out, for instance, that in the world of computing "run" contrasts with "load" or "program," "implement" is the opposite of "plan," "window" stands opposed to "the whole screen," and "go" is an alternative to "give a command." With such guidance, the novice can enter the metaphor's new semantic realm, look back at the pre-computer meaning, and grasp what Pico della Mirandola would call the "hidden alliances and affinities" between them.

The foregoing discussion may imply that only teachers and new users need to see both dimensions of computer metaphors—those specialists, who work only within Concept A, can safely pull up the bridge that carried them there. And indeed, professionals can understand each other well enough without remembering the semantic depth of their jargon.

On the other hand, a sensitivity to metaphor may benefit research and analysis in computer science itself. It may have been the metaphor in go that inspired a new kinesthetic device for cursor-movement—the mouse. In the same vein, the architectural metaphor underlying window could suggest other convenient interfaces between program and user: what about doors, or curtains? To take a more theoretical example, anyone seeking to understand structured programming can benefit from a study of the relationships among various kinds of "block," in and out of computer science. More broadly yet, unlimited challenges appear to those who take seriously the great metaphor implicit in such terms as intelligence, memory, and even computer (originally "person who computes"). The identity that such terms posit between a human Concept A and a mechanical Concept B raises some of the central and enduring questions of computer science.

Metaphor, writes George Whalley, may "prove to be the radical mode in which we correlate all our knowledge and experience" [10]. If computer science is indeed, in Marvin Minsky's words, "the long-awaited science of the relations between parts and wholes" [4, p. 393], its achievements depend in part on the vitality of the metaphoric imagination.

REFERENCES


Acronyms and initialisms are included among "neologisms" because like genuine neologisms, they are semantically opaque to the outsider.

Some terms from the economics and physics texts are listed here as "other" because they not fall into any of the three categories. They might be placed into a fourth category: "ordinary words made more precise." Like a "redefinition," the physicist's wave and the economist's labor are familiar from other contexts. Unlike redefinitions, however, such terms are not redefined by the technical field; instead, their definitions are quantified or operationalized. They present semantic problems of a different sort than those posed by redefinitions.

Information-processing text.

Neologisms. ADP; (Automatic Data Processing), Algorithm, Cybernetics, Millisecond, Multiplex

Technical terms. Binary, Closed subroutine, Data organization, Equipment, Peripheral, Flowchart, Linear programming, Magnetic core, Magnetic ink, Mathematical model, Numerical analysis, Octal, Problem description, Program, Rounding, Sequential control, Software, Terminal

General terms. And, Array, Attribute, Block, Byte, Central processing unit, Collate, Compile, Connector, Decision table, Document, Edit, Field, Fixed point, Gang punch, Head, Index, Input, Item, Key, Loop, Message, Operating system, Output, Pack, Patch, Real time, Relocate, Run, Storage, String, Switch, Table look-up, Transfer, Unconditional, Unit record, Verification

Programming text.

Neologisms. Get(f), Paren

Technical terms. Assignment operator, Cardinality, Compile-time error, Compound statement, Correct, Defensive programming, Efficiency, Empty statement, Evaluate, Exit condition, File window, Function, Iteration, List disposal, Mnemonic,
Operand, Predefined identifier, Program heading, Pseudorandom, Scale factor, Selection sort, Set operator, Solution space, Standard input, Standard output, State variable, Subrange, System defined, Top-down debugging, User-defined ordinal type

General terms. Actual (parameter), Allocate, Argument, Batch (computer or program), Bottom-up, Bubble sort, Comment, Constant, Curly brackets, Device, Directive, Driver, External, Field list, Flakey, Global, Handwave, Identifier, In, Indirect access, Input, Library, Location, Massage, Nesting, Number crunching, Output, Pop, Queue, Right thing, Root, Square brackets, Stepwise refinement, Structured walkthrough, Terminated, Transparency, Truth table, Undefined

Psychology text.

Neologisms. Androgens, Antidiuretic hormone, Cannon-Bard theory, Cochlea, Dendrite, Dichromatism, Endocrine gland, GSR (galvanic skin response), Gonads, Id, Interneurons, Introvert, Kinesthesia, LSD (lysergic acid derivatives), Mantra, Morpheme, Myelin sheath, PS4R method, Phenomenology, REMs (rapid eye movements), Saccade, Superego, Synapse, Volley principle

Technical terms. Achievement, Adrenalin, Age regression, Ambivalence, Anxiety hierarchy, Assertive training, Basal mental age, Behavior therapy, Biofeedback, Brain stem, Central core (of brain), Cerebral hemispheres, Chronological age (CA), Classical conditioning, Cognitive dissonance, Color blindness, Complex cell, Conditioned stimulus (CS), Conscience, Counterconditioning, Cumulative curve, Defense mechanism, Depolarization, Displaced aggression, Dominant gene, Eardrum, ego, Electroshock therapy, Equilibration, Evoked potential, Experimental method, Factor analysis, Fixation, Free association, Frustration-aggression hypothesis, Gene, Genetics, Hedonism, Heterosexuality, Hypnotic induction, Illusion, Independent variable, Inner ear, Intellectualization, Lateral fissure, Light adaptation, Localized functions, Memory span, Mental imagery, Middle ear, Narcotics, Negative reinforcement, Neurosis, Norm, Obsessive-compulsive disorder, Operant behavior, Oral stage, Ovarian hormones, Panic disorder, Parathyroid glands, Perceptual patternning, Personality disorders, Phobic disorder, Pitch, Polygenic traits, Positive reinforcer, Preoperational stage, Proactive interference, Psychiatrist, Psychodrama, Psychological motive, Psychophysics, Psychosurgery, Rapid eye movements, Reaction range, Recessive gene, Refractory phase, Reinforcement, Response, Retinal size, Scapegoat, Secondary sex characteristics, Self-perception, Semicircular canals, Septal area, Sex-role standards, Sibling, Smooth muscle, Sociology, Specific hunger, Stabilized retinal image, Stereoscopic vision, Stimulus-response (S-R) psychology, T-maze, Temperament, Test profile, Trait theory, Transsexual, Unconditioned response (UR), Variable, Visual area

Economics text.

Neologisms. Engel's Law, Laissez faire, Lorenz curve, Oligopsony

Technical terms. "Unfavorable" balance of trade, Administrative budget, Arbitration, Average cost, Balance of payments, Bank reserves, Boycott, Business cycles, Capital consumption, Capitalization, Compensatory fiscal policy, Constant costs, Countercyclical fiscal policy, Creditor nation, Crude death rate, Current assets, Debtor nation, Deflation, Deposit money, Depression, Diminishing marginal utility (law of), Disposable income, Dollar shortage, Economic growth, Excess reserves, Factors of production, Featherbedding, Fiscal policy, Fixed costs, Franchise, Full employment, Geometric progression, Gold standard, Gross investment, Holding company, Import quota, Innovation, Interlocking directorate, Least-cost combination, Liquidity preference, Marginal efficiency of capital, Marginal propensity to save, Marginal revenue product, Market period, Mediation, Mixed enterprise, Monopoly, National debt, Natural resources, Net national debt,
Normal profit, Operations research, Overhead costs, Patent, Personal tax, Primary deposits, Producers' good, Progressive tax, Propensity to save, Public utility, Pure monopoly, Real cost, Rediscount, Revenue, Saving function, Secondary boycott, Transfer payment, Turnover tax, Value added, Velocity of circulation

General terms. Acceptance, Closed shop, Index number, Short run, Time preference

Other. Assets, Consumption, Economics, Productivity, Supply, Technology

Physics text.

Neologisms. Adiabatic, Accelerometer, Ampere's law, Boltzmann constant, Boyle's law, Carnot cycle, Corpuscles, Einstein's principle of relativity, Fermat's principle of least time, Fraunhofer diffraction, Isotopes, Lenz's law, Lloyd's mirror, Lorentz transformation equations, Ohm meter, Oscilloscope, Pitot tube, Wheatstone bridge


General terms. Compliance, Dislocation, Disorder, Impulse, Overdamping, Virtual object, Wave packet

Other. Pitch, Power, Radiation, Tension, Wave, Work

Ecology text.

Neologisms. Allopatry, Aufwuchs, Benthos, Commensalism, Cryptophyte, Ecotone, Eolian deposit, Forb, Infauna, Loess, Mycorrhizae, Neuston, Paleolimnology, Periphyton, Pheromone, Saprophage

Technical terms. Productivity

***************************************************************************