Epistemological Pluralism and the Revaluation of the Concrete

by Sherry Turkle and Seymour Papert

Versions of this article appeared in the Journal of Mathematical Behavior, Vol. 11, No.1, in March, 1992, pp. 3-33; Constructionism, I. Harel & S. Papert, Eds. (Ablex Publishing Corporation, 1991), pp.161-191; and SIGNS: Journal of Women in Culture and Society, Autumn 1990, Vol. 16 (1). Sourced from http://www.papert.org/articles/EpistemologicalPluralism.html

Epistemological pluralism

The concerns that fuel the discussion of women and computers are best served by talking about more than women and more than computers. Women's access to science and engineering has historically been blocked by prejudice and discrimination. Here we address sources of exclusion determined not by rules that keep women out, but by ways of thinking that make them reluctant to join in. Our central thesis is that equal access to even the most basic elements of computation requires an epistemological pluralism, accepting the validity of multiple ways of knowing and thinking.

With this assertion we find ourselves at the meeting point of three epistemological challenges to the hegemony of the abstract, formal, and logical as the privileged canon in scientific thought. The first of these challenges comes from within feminist scholarship. Here, the canonical style, abstract and rule-driven, is associated with power and elitism, and with the social construction of science and objectivity as male.[1]

A second challenge comes from social scientists who have undermined the privileged status of canonical ways of knowing through their studies of scientific and mathematical practice. They show us how within laboratories there is a great deal of thinking that does not respect the canon and how "ordinary" people in their kitchens and workplaces make very effective use of a down-to-earth mathematical thinking very different from the abstract and formal math they were taught at school.[2]

Although few members of this community make a direct connection with feminism, there is a convergence of intellectual values -- a "revaluation of the concrete." These challenges to the dominant epistemology are intellectually assertive and politically self-conscious. A third challenge most often presents itself as neutral and technical. It is a challenge from within computation, as when the maverick Macintosh with its iconic interface made its bid against the established IBM perso-

nal computer. That the computer should be an ally in the revaluation of the concrete has a certain irony; in both the popular and technical cultures there has been a systematic construction of the computer as the ultimate embodiment of the abstract and formal. But the computer's intellectual personality has another side: Computers provide a context for the development of concrete thinking. When we look at particular cases of individuals programming computers, we see a concrete and personal approach to materials that runs into conflict with established ways of doing things within the computer culture. The practice of computing provides support for a pluralism that is denied by its social construction.[3]

Since the prevailing image of the computer is that of a logical machine, and since programming is seen as a technical and mathematical activity, the existence of anything but an analytic approach in this area makes a dramatic argument for pluralism. But the computer's most specific contribution to the critique of canonical styles depends on something more fundamental. The computer stands betwixt and between the world of formal systems and physical things; it has the ability to make the abstract concrete. In the simplest case, an object moving on a computer screen might be defined by the most formal of rules and so be like a construct in pure mathematics; but at the same time it is visible, almost tangible, and allows a sense of direct manipulation that only the encultured mathematician can feel in traditional formal systems (see Davis & Hersh, 1981; Papert, 1980a). The computer has a theoretical vocation: to bring the philosophical down to earth.[4]

While many can empathize when Carol Gilligan describes people making "contextual" moral decisions (you can cast yourself and acquaintances in the different roles) there is more of a problem when people try to get close to what it feels like to do science in a style that rejects standard notions of "objectivity."[5] Evelyn Fox Keller, describing such a style in the work of geneticist Barbara McClintock, notes that this is the "less accessible aspect" of a scientist's relationship to nature (Keller, 1985). We believe she is right. A personal appropriation of epistemological pluralism in science requires, at the limit, that we get close to the experiences of an Einstein or a McClintock or a Salk. But you can imagine yourself in the place of a programmer more easily than in the place of an Einstein. And when you yourself program (an activity within the reach of everyone), you can experience the degree to which your style of solving logical problems is very much your own.

In this chapter, we use the computer as an instrument for observing different styles of scientific thought and developing categories for analyzing them.[6] We find that, besides being a lens through which personal styles can be seen, it is also a privileged medium for the growth of alternative voices in dealing with the world of formal systems. After presenting cases in which the computer serves as an expressive medium for personal styles, we turn to this more speculative theme: As a carrier for pluralistic ideas, the computer holds the promise of catalyzing change, not only within computation but in our culture at large.

Personal appropriation

Consider Lisa, 18, a first-year Harvard student in an introductory programming course. Lisa had feared that she would find the course difficult because she is a poet, "good with words, not numbers." But after years of scorning teachers who had insisted that mathematics is a language, the computer has made Lisa ready to reconsider the proposition, and with it her characterization of herself as someone "bad at math." Lisa started well, surprised to find herself easily in command of the course material. But as the term progressed she reluctantly decided that she "had to be a different kind of person with the machine." She could no longer resist a pressure to think in ways that were not her own. She was in trouble, but her difficulty expressed a strength, not a weakness. Her growing sense of alienation did not stem from an inability to cope with programming but from her ability to handle it in a way that came into conflict with the computer culture she had entered. Lisa wants to manipulate computer language the way she works with words as she writes a poem. There, she says, she "feels her way from one word to another," sculpting the whole. When she writes poetry, Lisa experiences language as transparent; she knows where all the elements are at every point in the development of her ideas. She wants her relationship to computer language to be similarly transparent. When she builds large programs she prefers to write her own smaller "building block" procedures even though she could use prepackaged ones from a program library; she resents the latter's opacity. Her teachers chide her, insisting that her demand for transparency is making her work more difficult; Lisa perseveres, insisting that this is what it takes for her to feel comfortable with computers.

Two months into the programming course, Lisa's efforts to succeed were no longer directed towards trying to feel comfortable. She had been told that the "right way" to do things was to control a program through planning and black-boxing, the technique that lets you exploit opacity to plan something large without knowing in advance how the details will be managed. Lisa recognized the value of these techniques -- for someone else. She struggled against using them as the starting points for her learning. Lisa ended up abandoning the fight, doing things "their way," and accepting the inevitable alienation from her work. It was at this point that she called her efforts to become "another kind of person with the machine" her "not-me strategy," and began to insist that the computer is "just a tool." "It's nothing much," she said, "just a tool."

A classmate, Robin, is a pianist. Robin explains that she masters her music by perfecting the smallest "little bits of pieces" and then building up. She cannot progress until she understands the details of each small part. Robin is happiest when she uses this tried and true method with the computer, playing with small computational elements as though they were notes or musical phrases. Like Lisa, she is frustrated with black-boxing or using prepackaged programs. She too was told her way was wrong: "I told my teaching fellow I wanted to take it all apart, and he laughed at me. He said it was a waste of time, that you should just black box, that you shouldn't confuse yourself with what was going on at that low level. "

Lisa and Robin came to the programming course with anxieties about not "belonging" (fearing that the computer belonged to male hackers who took the machines and made "a world apart"),

and their experiences in it only served to make matters worse.[7] Although imaginative and carefully designed, the Harvard course taught that there is only one right way to approach the computer, a way that emphasizes control through structure and planning. There are many virtues to this computational approach (it certainly makes sense when dividing the labor on a large programming project), but Lisa and Robin have intellectual styles at war with it. Lisa says she has "turned herself into a different kind of person" in order to perform, and Robin says she has learned to "fake it." Although both women are able to get good grades in their programming course, they represent casualties of this war. Both deny who they are in order to succeed.

In their response and their defense, Lisa and Robin are not alone. In a survey, 37 women members of a local computer society included 17 who "changed their style to suit the fashion" when they began to interact with the "official" computer world. "I got my wrists slapped enough times and I changed my ways," says a college student for whom programming on her Macintosh was a private passion until she entered MIT. The cost of such "wrist slapping" is high: On an individual level, talent is wasted, self image eroded. On the social level, the computer culture is narrowed.

Such casualties are unnecessary. The computer can be a partner in a great diversity of relationships. The computer is an expressive medium that different people can make their own in their own way. But people who want to approach the computer in a "noncanonical" style are rarely given the opportunity to do so. They are discouraged by the dominant computer culture, eloquently expressed in the ideology of the Harvard course. Like Lisa and Robin, they can pass a course or pass a test. They are not computer phobic, they don't need to stay away because of fear or panic. But they are computer reticent. They want to stay away, because the computer has come to symbolize an alien way of thinking. They learn to get by. And they learn to keep a certain distance. One of its symptoms is the language with which they neutralize the computer as they deny the possibility of using it creatively. Recall how Lisa dismissed it as "just a tool."

These responses begin to show how discrimination in the computer culture takes the form of discrimination against epistemological orientations, most strikingly, against the approach preferred by Lisa and Robin.

Style as substance

Lisa and Robin share a preferred strategy for organizing work and a particular manner of identifying with computational objects. To capture the difference between them and their instructors, we describe a complex of attributes, an "approach" to knowledge, that encompasses several dimensions of difference. We isolate two approaches that serve its ideal types, theoretical prisms through which to see simplified projections of more complex realities. Lisa and Robin use a "soft" approach, and the instructors in their course are encouraging them to use a "hard" one.

Our culture tends to equate soft with feminine and feminine with unscientific and undisciplined. Why use a term, soft, that may begin the discussion of difference with a devaluation? Because to refuse the word would be to accept the devaluation. Soft is a good word for a flexible

and nonhierarchical style, open to the experience of a close connection with the object of study. Using it goes along with insisting on negotiation, relationship, and attachment as cognitive virtues. Our goal is the revaluation of traditionally denigrated categories. We do not argue that valuable thinking is not soft; we explore ways in which soft is a valid approach for men as well as women, in science its well as the arts.

Hard and soft are more than different approaches to computation. The phrase epistemological pluralism (rather than, for example, computational pluralism) underscores the generality of the issues. The computer forces general questions about intellectual style to reveal an everyday face. Even schoolroom differences in how children program computers raise issues that come up in a more abstract form in scholarly debates about scientific objectivity. The computer makes ideas about alternative scientific voices more concrete and therefore more appropriate because we can relate them, not only to the science of the scientists, but to our own thinking.

Observation of the soft approach to programming calls into question deeply entrenched assumptions about the classification and value of different ways of knowing. It provides examples of the validity and power of concrete thinking in situations that are traditionally assumed to demand the abstract. It supports a perspective that encourages looking for psychological and intellectual development within, rather than beyond, the concrete and suggests the need for closer investigation of the diversity of ways in which the mind can use objects rather than the rules of logic to think with.

The ideal typical hard and soft approaches are each characterized by a cluster of attributes. Some involve organization of work (the hards prefer abstract thinking and systematic planning; the softs prefer a negotiational approach and concrete forms of reasoning); other attributes concern the kind of relationship that the subject forms with computational objects. Hard mastery is characterized by a distanced stance, soft mastery by a closeness to objects.

Hard mastery is resonant with the logical and hierarchical elements of the traditional construction of "scientific method." Soft mastery has always had its place in the discourse of the arts and has always been glimpsed in the autobiographical writings of scientists. Only recently has it gained academic recognition as an integral element of scientific practice.[8]

When we say that hard and soft approaches are ideal types, we signal that individuals will seldom conform to either exactly, and that some will be so far from both that it is impossible to assign a type. In other words our contention is not that the attributes in a cluster are exactly correlated, but that each approach has internal coherency in the way that a stable culture is coherent. So for example, closeness to objects tends to support a concrete style of reasoning, a preference for using objects to think with, and a bias against the abstract formulae that maintain reason at a distance from its objects. Conversely, a distanced relationship with objects supports an analytic, rule- and plan-oriented style. Our theoretical conjecture is that degree of closeness to objects has developmental primacy; it comes first. The child forms a proximal or distant relationship to the world of things. The tendency to use the abstract and analytic or concrete and negotiational style of thinking follows.

But although closeness to objects favors contextual and associational styles of work, it does not

exclude the possibility of using a hierarchical one. Planning is not always an expression of personal style. It can be acquired as a skill, sometimes because it is needed to get a job done, sometimes as a facade to hide rather than express individuality.

Thus, the elements of each cluster are not invariably associated with each other; still less are they invariably associated with gender.[9] But in our observations of people learning to program we have found an association between gender and approach to programming. When people are free to explore programming without preconceptions about the "right" way to do it, more women use soft approaches and more men hard approaches, although many men are alienated from the dominant engineering style and many women work creatively within it.[10]

Using clinical methods inspired by the Piagetian and psychoanalytic traditions, we built up case studies of children using computers in grade-school settings where they were encouraged to explore programming without preconceptions about the "right way" to go about it. We took 40 cases for which we had material both on individual personality and programming style. Of 20 girls, 14 favored the soft approach; of 20 boys, there were four who followed this route. In case material on college students taking a first programming course, of the 15 women, nine were soft style programmers; of 15 men, four. What we say in this chapter about gender, programming, and intellectual style is based on the analysis of these cases. But we believe that what is most important is not any statistical association between gender and programming styles, but what lies behind the styles and behind the resistance of our intellectual culture to recognize and facilitate them both. In our culture, those who use hard approaches don't simply share a style, they constitute an epistemological elite.

Here, we focus on the soft approach; the canonical style is well known and well defended. But implicitly, our discussion of soft approaches is a discussion of hard ones; it contributes to the deconstruction of the latter as the way to do things. It also situates it: the supervaluation of the hard approach owes much of its strength within computation to the support it gets in other intellectual domains. To state simplistically a position we shall elaborate in the following pages: "Hard thinking" has been used to define logical thinking. And logical thinking has been given a privileged status that can be challenged only by developing a respectful understanding of other styles where logic is seen as a powerful instrument of thought but not as the "law of thought." In this view, "logic is on tap, not on top."

It is beyond the scope of this essay to spell out the multiple relationships between our definition of "soft" and the large body of feminist writings on intellectual approaches. But we use the work of Gilligan and Keller to bring out the two most striking constituent elements of the soft approach. The negotiational and contextual element, which we call bricolage, recalls Gilligan's material on moral reasoning. Then we look "beyond bricolage" to a second element of the soft approach, underscored by Keller in her work on Barbara McClintock. This is a style of relating to objects, be they physical objects such as gears or chromosomes, or conceptual objects such as the elements of programming. Whereas bricolage is negotiational, this aspect of the soft approach is proximal, a closeness to objects.

Bricolage

Levi-Strauss used the idea of bricolage to contrast the analytic methodology of Western science with what he called a ”science of the concrete” in primitive societies.[11] The bricoleur scientist does not move abstractly and hierarchically from axiom to theorem to corollary. Bricoleurs construct theories by arranging and rearranging, by negotiating and renegotiating with a set of well-known materials.

If we take Levi-Strauss's description of the two scientific approaches as ideal types and divest them of his efforts to localize them culturally, we can see both in how people program computers. For some people, what is exciting about computers is working within a rule-driven system that can be mastered in a top-down, divide-and-conquer way. Their structured ”planner's” approach, the approach being taught in the Harvard programming course, is validated by industry and the academy. It decrees that the ”right way” to solve a programming problem is to dissect it into separate parts and design a set of modular solutions that will fit the parts into an intended whole. Some programmers work this way because their teachers or employers insist that they do. But for others, it is a preferred approach; to them, it seems natural to make a plan, divide the task, use modules and subprocedures.

Lisa and Robin offer examples of a very different style. They are not drawn to structured programming; their work at the computer is marked by a desire to play with the elements of the program, to move them around almost as though they were material elements -- the words in a sentence, the notes on a keyboard, the elements of a collage.

While hierarchy and abstraction are valued by the structured programmers' ”planner's” aesthetic, bricoleur programmers, like Levi-Strauss's bricoleur scientists, prefer negotiation and rearrangement of their materials. The bricoleur resembles the painter who stands back between brushstrokes, looks at the canvas, and only after this contemplation, decides what to do next. Bricoleurs use a mastery of associations and interactions. For planners, mistakes are missteps; bricoleurs use a navigation of midcourse corrections. For planners, a program is an instrument for premeditated control; bricoleurs have goals but set out to realize them in the spirit of a collaborative venture with the machine. For planners, getting a program to work is like ”saying one's piece”; for bricoleurs, it is more like a conversation than a monologue.

We introduced bricolage through the programming styles of college students Lisa and Robin. To consider it in finer detail, we look at the work of two 9-year-olds. In the spirit of Jean Piaget, we find that children's thinking often allows particularly transparent access to processes that extend far beyond childhood.

Alex, 9 years old, a classic bricoleur, attends the Hennigan Elementary School in Boston, the scene of an experiment in using computers across the curriculum. There, students work with Logo programming and computer controlled Lego construction materials. The work is both frequent enough (at least an hour a day) and open-ended enough for differences in styles to emerge.

When working with Lego materials and motors, most children make a robot walk by attaching wheels to a motor that makes them turn. They are seeing the wheels and the motor through

abstract concepts of the way they work: the wheels roll, the motor turns. Alex goes a different route. He looks at the objects more concretely; that is, without the filter of abstractions. He turns the Lego wheels on their sides to make flat "shoes" for his robot and harnesses one of the motor's most concrete features: the fact that it vibrates. As anyone who has worked with machinery knows, when a machine vibrates it tends to "travel," something normally to be avoided. When Alex ran into this phenomenon, his response was ingenious. He doesn't use the motor to make anything "turn," but to make his robot (greatly stabilized by its flat "wheel shoes") vibrate and thus "travel." When Alex programs, he likes to keep things similarly concrete.

Learners are usually introduced to Logo programming through the "turtle," an icon on a computer screen which can be commanded to move around the screen and leave a trace as it goes. So, for example, the turtle can be told to move forward a hundred steps and turn ninety degrees with the commands FORWARD 100 RIGHT 90. Four such commands would have the turtle drawing a square. Programming occurs when a set of commands, such as REPEAT 4 [FORWARD 100 RIGHT 90], are defined as a procedure: TO SQUARE. Alternatively, a subprocedure TO SIDE might be defined and repeated four times.

Alex wanted to draw a skeleton. Structured programming views a computer program as a hierarchical sequence. Thus, a structured program TO DRAW SKELETON might be made up of four subprocedures: TO HEAD, TO BODY, TO ARMS, TO LEGS, just as TO SQUARE could be built up from repetitions of a subprocedure TO SIDE. But Alex rebels against dividing his skeleton program into subprocedures; his program inserts bones one by one, marking the place for insertion with repetitions of instructions. One of the reasons often given for using subprocedures is economy in the number of instructions. Alex explains that doing it his way was "worth the extra typing" because the phrase repetition gave him a "better sense of where I am in the pattern" of the program. He had considered the structured approach but prefers his own style for aesthetic reasons: "It has rhythm," he says. In his opinion, using subprocedures for parts of the skeleton is too arbitrary and preemptive, one might say abstract. "It makes you decide how to divide up the body, and perhaps you would change your mind about what goes together with what. Like, I would rather think about the two hands together instead of each hand with the arms."[12]

In his own way, Alex has resisted the pressure to believe the general superior to the specific or the abstract superior to the concrete. For Alex, thinking about hands as a subset of arms is too far away from the reality of real hands, just as taking a motor that was most striking as a vibrating machine and using it to turn wheels in the standard fashion was too far away from the real motor he had before him. While the structured programmer starts with a clear plan defined in abstract terms, Alex lets the product emerge through a negotiation between himself and his material. In cooking, this would be the style of chefs who don't follow recipes but a series of decisions made as a function of how things taste. Or we might think of sculptors who let themselves be guided by the qualities of the stone that reveal themselves as the work progresses.

The turtle that Alex used to draw a skeleton is the best known Logo object, but there are others. For example, sprites are turtles that can be set in motion. Once you give a sprite a speed and a heading, it moves with that state of uniform motion until something is done to change it, just like

an object obeying Newton's first law.

Anne, whose favorite hobby is painting, has become expert at using sprites in programs that produce striking visual effects.[13] In one, a flock of birds (each of them built from a sprite) flies through the sky, disappears over the horizon, and reappears some other place and time. If all the birds were red, then it would be easy to make them disappear and reappear. The command SETCOLOR :INVISIBLE would get rid of them and SETCOLOR :RED would make them reappear. But Anne wants the birds to have different colors, and so making the birds reappear with their original color is more complicated.

A classical method for achieving this end calls for an algebraic style of thinking: You make the program store each bird's original color as the value of a variable, then you change all colors to invisible and recall the appropriate variable when the bird is to reappear. Anne knows how to use this algorithmic method, but prefers one that allows her to turn programming into the manipulation of familiar objects. As Anne programs, she uses analogies with traditional art materials. When you want to hide something on a canvas, you paint it out, you cover it over with something that looks like the background. Anne uses this technique to solve her programming problem. She lets each bird keep its color, but she makes her program "hide" it by placing a screen over it. Anne designs a sprite that will screen the bird when she doesn't want it seen, a sky-colored screen that makes the bird disappear. Anne is programming a computer, but she is thinking like a painter.

"Thinking like a painter" does not prevent Anne from making a significant technical innovation in the context of her fourth-grade computer culture. She is familiar with the idea of using two sprites to form a compound object. Her classmates and teachers have always done this by putting the sprites side by side. Anne's program is like theirs in using two sprites, one for the screen, one for the bird. But she places them on top of each other so that they occupy the same space. Instead of thinking of compound objects as a way of getting a picture to be bigger, she thinks of compound objects as a way of getting sprites to exhibit a greater complexity of behavior, an altogether more subtle concept.

Thus, Anne's level of technical expertise is as dazzling in its manipulation of ideas as in its visual effects. She has become familiar with the idea of data structures by inventing a new one -- her "screened bird." She has learned her way around a set of mathematical ideas through manipulating angles, shapes, rates, and coordinates in her program. As a bricoleur, her path into this technical knowledge is not through structural design, but through the pleasures of letting effects emerge.

In describing bricoleur programmers, we have made analogies to sculptors, cooks, and painters. Bricoleurs are also like writers who don't use an outline but start with one idea, associate to another, and find a connection with a third. In the end, an essay "grown" through negotiation and association is not necessarily any less elegant or easy to read than one filled in from an outline, just as the final program produced by a bricoleur can be as elegant and organized as one written with the top-down approach.

Anne's case makes it clear that the difference between planners and bricoleurs is not in quality of product, it is in the process of creating it. As in the case of Alex, Anne does not write her program in "sections" that are assembled into a product. She makes a simple working program and

shapes it gradually by successive modifications. She starts with a single black bird. She makes it fly. She gives it color. Each step is a small modification to a working program that she has in hand. If a change does not work, she undoes it with another small change. She "sculpts." At each stage of the process, she has a fully working program, not a part but a version of the final product.

Anne is perfectly capable of producing a program with well-delineated subprocedures, although her way of creating them has little in common with the planner.[14] Devotees of structured programming would frown on Anne's style. From their point of view, she should design a computational object (for example, her bird) with all the required qualities built into it. She should specify, in advance, what signals will cause it to change color, disappear, reappear, and fly. One could then forget about "how the bird works." In engineer's jargon, it could be treated as a black box. Anne's work dramatizes the feature of bricolage that was so salient for Lisa and Robin: the desire for transparency.

With a structured programming style, one usually does not feel comfortable with a construct until it is thoroughly black-boxed, with both its inner workings and all traces of the perhaps messy process of its construction hidden from view. Many programmers feel a sense of power when they use black-boxed programs, perhaps because of the thought that others might take them up exactly as frozen.

But black-boxing makes other programmers nervous rather than exultant. Anne did not want to package her constructs into opaque containers. Like Lisa and Robin, she enjoys keeping open the possibility of renegotiating their exact form. And this means staying in touch with that form at all times. When programming, bricoleurs tend to prefer the transparent style, planners the opaque, but the program's authorship is a critical variable in this preference. Planners want to bring their own programs to a point where they can be black-boxed and made opaque, while bricoleurs prefer to keep them transparent. But when dealing with programs made by others, the situation is reversed. Now, the bricoleurs are happy to get to know a new object by interacting with it, learning about it through its behavior the way you would learn about a person, while the planners usually find this intolerable. Their more analytic approach demands knowing how the program works with a kind of assurance that can only come from transparent understanding, from dissection and demonstration.

Do programmers "graduate" from bricolage when they develop greater expertise? Will Anne become a structure programmer in junior high? Our observations suggest that, with experience, bricoleurs reap the benefits of their long explorations, so that they may appear more "decisive" and like planners when they program on familiar terrain. And of course, they get better at "faking it." But the negotiating style resurfaces when they confront something challenging or are asked to try something new. Bricolage is a way to organize work. It is not a stage in a progression to a superior form. Indeed, there is a culture of adult programming virtuosos, the hacker culture, that would recognize many elements of the bricolage style as their own. And interviews with graduate students in computer science turned up highly skilled bricoleurs, most of them aware that their style was "countercultural."

In the case of computation, the existence of the countercultural style challenges the idea of one

privileged, "mature" approach to problems. This challenge is supported by countercultural styles in other domains, for example, those observed by Gilligan in the domain of moral reasoning.

Gilligan's work presents us with two moral voices. We can hear both in children's responses to classical examples of moral dilemmas. When confronted by the story of Heinz, who must decide whether to steal a drug to save a life, 11-year-old Jake sees the dilemma as "sort of like a math problem with humans" (Gilligan, 1982, p. 26). He sets it up as an equation and arrives at what he believes is the universal response: Heinz should steal the drug because a human life is worth much more than money. Eleven-year-old Amy takes an approach in which we see elements of bricolage. While Jake accepted the abstractly given problem as a quantitative comparison of two evils, Amy looks at the problem setting in concrete terms, breaks the restrictive formal frame of the given problem, and introduces a set of new elements. These elements include the druggist as a concrete human being who probably has a wife of his own and feelings about her. Amy proposes that Heinz talk things over with the druggist, who surely will not want anyone to die.

In Gilligan's description of Jake, justice was like a mathematical principle. It resembled the structured programmer's black box. To solve a problem, you set up the right algorithm, the right box, you crank the handle, and the answer comes out. Amy's style of reasoning required her to stay in touch with the inner workings of her arguments, with the relationships and possibly shifting alliances of a group of actors. Amy's resemblance to the programmers Alex and Anne is striking. They are all negotiators, stay close to their materials, and require transparency as they arrange and rearrange them. Despite Anne's high level of achievement, theorists of structured programming would criticize her style for the same kinds of reasons that Lawrence Kohlberg would classify the impressively articular Amy at a lower intellectual level than Jake. In both cases, criticism would center on the fact that neither of the two young women is prepared to take the final step to abstraction.

Kohlberg saw moral development as a sequence of stages that move from judging rightness by one's immediate feelings to judging rightness by the application of absolute principles. Between egocentrism and the Kantian imperative lie intermediate stages of reasoning based on balancing and assessing the consequences of actions for individuals. Gilligan finds many adult women speaking as did Amy. In Kohlberg's terms, they are "blocked" at this intermediate level: Instead of looking to universal principles in making their decisions, they consider concrete situations. Gilligan uses her observations to reject Kohlberg's theory, particularly its positing of a determinate end-point to development.[15] If one branch of the development of moral reasoning moves towards the primacy of "justice," of the formal and analytic, Gilligan insists on equal respect for a different branch of development which leads toward increasingly sophisticated ways of thinking about morality in concrete terms of care through relationship and connection.

In making the analogy between Amy and Anne, we shift the emphasis of Gilligan's analysis of the "different voice." Is her work about morality or epistemology? Gilligan is certainly concerned with both morality and epistemology when she says: "[for women] the moral problem arises from conflicting responsibilities rather than from competing rights and requires for its resolution a mode of thinking that is contextual and narrative rather than formal and abstract" (Gilligan, 1982,

p. 19). But her language expresses a priority, a primary concern with the character of the morality which, as she says, requires a certain mode of thinking. Our concern is with the mode of thinking.

Gilligan's priority shows itself in recent writing where she redescribes Kohlberg's theory as being about only one side of moral reasoning. In this view, Kohlberg is talking about justice, thus leaving the other side of morality, namely care, to her (Gilligan, 1988).

This compromise, which splits off the content of moral judgment, blunts the force of Gilligan's observations as a challenge to something more general than moral reasoning. Kohlberg's theory of the development of moral judgment mirrors Piaget's theory of the development of intelligence per se. Both express the value-laden perspective on intellectual growth that has dominated Western philosophy. Piaget sees a progression from egocentric beginnings to a final, "formal stage" when propositional logic and the hypothetico-deductive method "liberate" intelligence from the need for concrete situations to mediate thinking (Piaget & Inhelder, 1958). In this vision, mature thinking is abstract thinking. We disagree: for us, formal reasoning is not a stage, but a style.

Although Piaget would place the "concrete" Anne squarely in the preformal stage, her level of achievement undermines standard assumptions about the privileged status of the analytic and formal. And it undermines standard assumptions about the "objective." There is little distance between Anne and her objects. This aspect of Anne's work -- her close, almost tactile involvement with the sprites -- enables us to make a bridge between styles of programming and styles, this time, not of moral discourse, but of doing science. The fact of diverse styles of expert programming supports the idea that there can be different but equal voices even where the formal has traditionally appeared as almost definitionally supreme: logic, mathematics, and the "hard" sciences. And this aspect of Anne's work bring us to the second constituent element of the soft approach. We go "beyond bricolage" to the question of closeness to the object.

Beyond bricolage: closeness to the object

There is a tradition of scientific epistemology that sees the essence of science in objectivity and the essence of objectivity in a distanced relationship with the object of study. Feminist scholars have related this notion of objectivity to the construction of gender: objectivity in the sense of distancing the self from the object of study is culturally constructed as male, just as male is culturally constructed as distanced and objective. In a moving case study of a scientist at work, Keller demonstrates that this is not the only possible stance. She sees another in the work of geneticist Barbara McClintock, who talked about her relationship to objects of scientific study as one of proximity rather than distance. For McClintock, the practice of science was essentially a conversation with her materials. "Over and over again," says Keller, McClintock "tells us one must have the time to look, the patience to 'hear what the material has to say to you,' the openness to 'let it come to you.' Above all, one must have a 'feeling for the organism.' " [16]

McClintock said that the more she worked with neurospora chromosomes (so small that others had been unable to identify them), "the bigger [they] got, and when I was really working

with them I wasn't outside, I was down there. I was part of the system. I actually felt as if I were right down there and these were my friends . . . As you look at these things, they become part of you and you forget yourself" (Keller, 1983, p. 117). McClintock came into increasing conflict with the formal, "hard" methods of molecular biology. McClintock was recognized by the scientific establishment, indeed she was awarded the Nobel Prize, only when the formal approach came independently and much later to conclusions that she had derived from her "softer" investigations.

When we study programmers at work we see differences reminiscent of the two approaches to genetics. Alex and Anne relate to computational objects much as McClintock related to chromosomes, while many of their peers, like the mainstream molecular biologists, take a more distant approach. Anne psychologically places herself in the space of her objects. She experiences her screens and birds as tangible, sensuous, and tactile. She is down there, in with the sprites, playing with them like objects in a collage. Or consider the computer science graduate student Lorraine, who explains how she uses "thinking about how the program feels like inside" to break through difficult problems. "For appearances sake," she wants to "look like I'm doing what everyone else is doing, but I'm doing that with only a small part of my mind. The rest of me is imagining what the components feel like. It's like doing my pottery." This is in sharp contrast to structured programmers who use their favorite device of black-boxing as a way to maintain distance. The idea of the black box, designed not to be touched, mediates between the structured (planning) style of organizing work and their relationship to computational objects. Structured programmers are not among the sprites, they act on the sprites.

An example from outside of computation clarifies the role of identification with objects as a way of appropriating formal systems. Children using gears illustrate how closeness to objects allows alternatives to abstract thinking.

At the Hennigan School where we met Alex, many students work as he does with a Lego construction kit with which they can build mechanisms and write computer programs to make them function. Sooner or later in building their objects, the children run into the need for gears.[17]

The motors in the construction set turn at a high speed with low torque. A car built by attaching these motors directly to the wheels will go very fast but will be so underpowered that the slightest slope or obstruction will cause it to stall. The solution to the problem with Lego cars is the same as that adopted by designers of real cars: use gears. But in order to use them effectively, children need to understand something about gear ratios. Contrary to their teachers' expectations, the girls in this project did extremely well, both in the quality of their work with the gears and in their performance on a test of underlying principles.[18]

If a small gear drives a larger gear, the larger gear will turn more slowly and with greater torque. It is the relative and not the absolute size of the two gears that counts. But when we interview children, we find that some of them reason as if the size of only one gear matters, as if they were following a set of rules such as "large gears are slow and strong" and "small gears are fast and weak." Without the notion of relative size, such rules fail. Other children, among them the girls who excelled, are less articulate and more physical in their explanations. They squirm and twist their bodies as they try to explain how they figure things out. And they get the right answer.

Theorists who look at intellectual development as the acquisition of increasingly sophisticated rules would say that children run into problems if the rules they have built are not yet good enough.[19] But armed with the idea of "closeness to objects," we can consider a different kind of theory. Perhaps the girls who did so well did not have better rules, but a tendency to see things in terms of relationships rather than properties. Perhaps the girls had easy access to a style of reasoning which allowed them to imagine themselves "inside the system." They used a relationship to the gears to help them think through a problem.

This kind of "reasoning from within" may not be adequate for all problems about gears, but for the kind of problem encountered by the children in our project, it was not only adequate, but much less prone to the errors produced by a too-simple set of rules. Relational thinking puts you at an advantage: You don't suffer disaster if the rule isn't exactly right.

This way of thinking about girls and gears is supported by the hypothesis, familiar in recent writing about women, that boys are more comfortable with boundaries and girls with attachments.[20] This notion, drawn from accounts of development in the psychoanalytic tradition of object relations, stresses that for girls, identity formation takes place in a context of greater continuity than for boys.[21] The girl's sexual identification is with the mother, with whom she is encouraged to maintain a close relationship. Girls don't need to define themselves through a denial of the early, closely bonded relationship with the mother to the same extent as boys. They grow up with a stronger basis for experiencing the needs and desires of another as their own. Since girls do not have to renounce the pleasures of attachment to the mother as sharply as do boys, it is easier for them to play with the pleasures of closeness to other objects as well.

For boys, the separation from the mother is sharper, because in a certain sense, it happens twice, first in the rupture of the earliest bonded relationship, then in the course of the Oedipal struggle. The double separation translates into a lifelong tendency to be most comfortable with clear boundaries between self and nonself. It makes distanced, "objective" relationships feel like safe, approved ground.

The contemptuous comment of one fourth-grade boy who overheard a classmate talking about "being a sprite" when he programs can be interpreted from this point of view. "That's baby talk," he said. "I am not in the computer. I'm just making things happen there." The remark reflects an insistence on boundaries and the development of a world view that will fall easily into line with the canonical, objective science whose male meanings Keller has delineated.

The object relations school of psychoanalysis focuses on the way development progresses by a process of internalization of the things and people of the world. They come to live within us; they become our objects to think with. When psychoanalysts talk about "objects," they usually mean people.[22] Keller, in her work on McClintock, has explicitly extended the idea of closeness to the object to elaborate a theory of relationships to nature. Here we further extend this idea to relationships with specific artifacts. In doing so, we find ourselves addressing questions more familiar in discussions about relationships between people than between people and things. What kinds of individuals choose to manipulate or make what kinds of objects, and what kinds of relationships follow? From this perspective, it is not enough to ask whether individuals "like" or "don't like" to

program, because that puts the question on too high a level of generalization. ”Liking” to program depends on forging an appropriate relationship with a computational object that ”fits.”

The choice of a computational object that fits has several elements. First is the choice among objects offered by the system. Even people who like to be close to objects don't all like to be close to the same ones. For example, in the version of Logo used by Anne there was a choice between sprites and turtles. Some prefer the turtle, its static nature, the fineness in the way it draws. For others, these same qualities are reasons to reject the turtle as constraining, even unpleasant. They prefer the sprites, which move with flash and speed.

Second, when an object has been chosen, it can be thought of in different ways. As computational objects, turtles and sprites stand on the boundary between the physical and the abstract. In some ways both are like physical objects. You can see them, move them, put one on top of another. But at the same time, they are abstract and mathematical. Ambivalent in their nature, computational objects can be approached in different ways. Hard-approach programmers treat a sprite more like an abstract entity -- a Newtonian particle -- while soft-approach programmers treat it more like a physical object -- a dab of paint or a cardboard cutout.

Because of this ambivalence, computational objects offer a great deal to those whose approach requires a close relationship to an object experienced as tactile and concrete. Computational objects offer a physical path of access to the world of formal systems. Some people are comfortable with mathematical experiences that manipulate symbols on quadrille-ruled paper. But for many the ambivalent nature of computational objects means quite simply a first access to mathematics.[23]

A third dimension of difference in people's choice of computational object has to do with differences in the degree to which these objects are anthropomorphized. The anthropomorphization extends from the computational objects (”That sprite doesn't want to do what I tell it now”) to the computer itself. Anne, like many soft-approach programmers, has an easier time working with the computer if she anthropomorphizes it. She has no doubt that computers have psychologies: They ”think” but can't really have ”emotions.” She believes, however, that the computer has preferences. ”He would like it if you did a pretty program.” And when it comes to technical things, Anne assumes the computer has an aesthetic: ”I don't know if he would rather have the program be very complicated or very simple.”

We know and she knows that the computer is ”just a machine.” But those who want to treat it in certain ways ”as if” it were a person are able to see the machine as sufficiently alive for it to serve as a companion, if only a limited one. When the computer ”moves the queen” in a game of chess, it tempts us to think of it as having intentions. And programs within a computer system interact with each other in a way that supports models of the computer as composed of ”agents” in communication. Anthropomorphization, both of a computer system and its parts, does not follow from lack of technical expertise. Computer scientists talk about a concept such as recursion with anthropomorphic metaphors: one agent ”calls up” another, ”wakes up” another, and ”passes on a job.” They sometimes even refer to the agents within a computer system as citizens of a ”society of mind” (Minsky, 1987; Papert, 1980c).

Very young children are in fact uncertain whether computers should be counted as alive or not

alive, and argue the question hotly, debating the computer's aliveness on the basis of its psychology, intentions, consciousness, and feelings (Turkle, 1984, esp. chap. I). By age 10, most are sure that the computer is not actually alive. But at this point, some children, like Anne, continue to behave with and talk about the computer as if it were sentient. They brag that it is helping them or complain that it isn't. In this, they are not showing confusion about biology. They do not think that the computer is alive the way an animal is. But it has a "kind of life," the kind of life appropriate to a computer. This is a psychological life.

Other children have a very different reaction. Once they are no longer perplexed by whether the machine might actually be biologically alive, they shy away from anthropomorphization. When they complain about the computer they do so in objective terms: It is too slow, it doesn't have enough memory. Talking about the computer usually means "talking shop" about technical details.

Lise Motherwell, a researcher at the Hennigan School, did an intensive study of eight fifth-grade students in a computer-rich classroom (Motherwell, 1988). She found she could capture children's stances towards the anthropomorphization of the computer by distinguishing two styles: relational and environmental. Relational children treat the computer as much like a person as they can get away with, while environmental children treat it like a thing. Three out of the four girls in her study were relational; three out of the four boys environmental. Motherwell's research supports our observation that children who anthropomorphize the computer are no less technically sophisticated than those who do not. The degree of anthropomorphization does not reflect expertise but psychological approach. Motherwell's research also supports our association of programming style and gender.

Looking at Motherwell's data through our theoretical prism, it is as though, once they have placed the computer in the not-alive category, the boys tend to settle with relief into treating it as a thing. This helps them to appropriate it through a relationship that involves distance, objectivity, and control. And it is as though the girls, once having settled the question of biological aliveness, get more comfortable with the machine by making it an interactive partner. In the computer they have found something in the domain of formal systems to which they can relate with informality.

The conventional route into formal systems, through the manipulation of abstract symbols, closes doors that the computer can open. The computer, with its graphics, its sounds, its text and animation, can provide a port of entry for people whose chief ways of relating to the world are through movement, intuition, visual impression, the power of words and associations. And it can provide a privileged point of entry for people whose mode of approach is through a close, bodily identification with the world of ideas or those who appropriate through anthropomorphization. The computational object, on the border between the idea and a physical object, offers new possibilities.

Closeness and conflicts

Different intellectual perspectives provide suggestive hypotheses about why women tend to adopt the approaches that we have clustered under the rubric soft. Some perspectives, such as the psychoanalytic account, place the roots of difference at an early stage of human development. If the earliest and most compelling experiences of merging are with the mother, the process of differentiation takes on gender meanings. Experiences where boundaries are not clear are associated with something female. Differentiation and delineation are male.

The psychoanalytic focus on early experience does not necessarily undermine a more sociological perspective, where the emphasis is on our culture's sharp gender division of parenting roles and on the very different socializations of men and women. As a birthday gift, a boy receives toy tanks and soldiers; a girl receives dolls, presented to her, not as objects to command, but as children to nurture. In our culture, girls are taught negotiation, compromise, and the capacity for intimacy as social virtues, while models of male behavior stress decisiveness, cool impartiality, and the imposition of will. It would not be surprising if women felt more comfortable, more "themselves," with negotiation and compromise among elements of thought, and men preferred to make decisive plans and impose principles on a separate reality.

From its very foundations, objectivity in science was metaphorically engaged with the language of male domination and female submission. Francis Bacon used the image of the male scientist putting the female nature "on the rack" (Keller, 1985, pp. 33 ff.; Merchant, 1980; Haraway, 1979, pp. 206-237). Objectivity has been constructed, not only in terms of the distance of the knower from nature, but in terms of an aggressive relationship towards it (or rather towards her). And from its very foundations, objectivity in science has been engaged with the language of power, not only over nature, but over people and organizations. Such associations have spread beyond professional scientific communities; aggression has become part of a widespread cultural understanding of what it means to behave in a scientific way. Its hard methods are expected to involve "demolishing" an argument and "knocking it down" to size. Here the object of the blows is not a female nature but a male scientific opponent. If science is first a rape, it is then a duel.

In either case, it is not surprising that many women feel uncomfortable both with science and with ways of thinking that have been associated with it. The traditional discourse of computation has not been exempt from these connotations. Programs and operating systems are "crashed" and "killed." We write this chapter on a computer whose operating system asks if it should "abort" an instruction it cannot "execute." This is a style of discourse that few women fail to note.

Such observations about language, power, and the genderization of scientific discourse suggest pressures that push women to look for alternative approaches to knowing, and this means pressure to adopt soft approaches. A psychoanalytic perspective might suggest that women have a predisposition towards a soft approach. And then, the social construction of science reinforces this preference. Science waves its flag as "hard" in a way that repels women. This means that, in our culture, women are too often faced with the not necessarily conscious choice of putting themselves at odds either with the cultural meanings of being a scientist or with the cultural constructions of

being a woman.

Such choices do not have to be made in an all-or-nothing way. The "not-me" strategy we mentioned earlier anticipated a phenomenon of costly partial surrender. Women who enter centers of power give eloquent testimony, not only to the pressure, but to the seduction of having achieved the right to use the male discourse that predominates there. And they also speak to the conflict that this engenders, a kind of conflict that considerably complicates our story of how women appropriate technology.

We have suggested that many women have a preference for attachment and relationship with technological objects as a means of appropriating them -- but we have also pointed to the association of these objects with a construction of the male that stresses aggression, domination, and competition. This construction of technology may lead to a conflict between a close encounter with technology and women's image of themselves as women. Such a conflict is apparent when we look at women and computers.

When women neutralize the computer as "just a tool," it is more than a way of withdrawing because of a lack of authenticity in style of approach. Insisting that the computer is just a tool is one way to declare that what is most important about being a person (and a woman) is incompatible with close relationships to technology.

When Lisa first found herself doing well in her programming course, she found it "scary," because she felt she needed to protect herself from the idea of "being a computer science type." In high school, Lisa saw young men around her turning to computers as a way to avoid people: "They took the computers and made a world apart." Lisa describes herself as "turning off" her natural abilities in mathematics that would have led her to the computer. "I didn't care if I was good at it. I wanted to work in worlds where languages had moods and connected you with people." And although Robin had gone through most of her life as a musician practicing piano eight hours a day, she too had fears about "guys who established relationships" with the computer. "To me, it sounds gross to talk about establishing a relationship with the computer. I don't like establishing relationships with machines. Relationships are for people."

In the vehemence with which many women insist on the computer's neutrality, on its being nothing more than a mere tool, there is something more than alienation of culture and style. Many women are fighting against an experience of the computer as psychologically gripping. They are fighting against an element of their soft approach. For some, it can be because they want to "belong" to the dominant computer culture. Lorraine, who programs by imagining what "the components feel like," ends her description of her programming style by adding, "Keep this anonymous. I know it sounds stupid." But for others, their experience of closeness to the object is a source of conflict.

When Lisa began programming, she saw herself as communicating with the computer, but the metaphor soon distressed her. "The computer isn't a living being, and when I think about communicating with it, well, that's wrong. There's a certain amount of feeling involved in the idea of communication, and I was looking for that from the computer." She looked at it, and she frightened herself. "It was horrible. I was becoming involved with a thing. I identified with how the computer

was going through things."

Lisa, like the soft-approach programmer Anne, placed herself in the space of the computational objects she worked with and was prone to anthropomorphization, responding to the computer as though it had (at least) an intellectual personality. In Lisa's case, her own style came to offend her. As a programmer with a soft approach to the discipline, she rebelled against where her style had led her, because it had led her to what she experienced as a too-close relationship with a machine.

Carol Gilligan talks about the "hierarchy and the web" as metaphors to describe the different ways in which men and women see their worlds (Gilligan, 1982, p. 62). Men see a hierarchy of autonomous positions. Women see a web of interconnections among people. Men can be with the computer, content that it leaves them alone, even isolated, within a larger organization. When women see computers demanding separation from others, they perceive the machines as dangerous. They use metaphors from their programming classes to frame a view of people as what computers are not. So, for example, Robin says that people have "great flashes of abstract thought without any logical sequence before it. If you tried to do that with a computer, it would tell you it's a system error or illegal!" Lisa boils down what computers can't do to a starker form: They cannot love.

I suppose if you look at the physical machinery of the computer mind, it is analogous to the human mind. But the saving grace, the difference, is emotion. That's the line you can draw. That's where you say, "This thing may be able to do something like thinking, but it can't love anybody."

The computer presence has provoked a "romantic reaction" in our culture.[24] As people take computers seriously as simulated mind, many are in conflict with the mechanistic image that is reflected back to them in the mirror of the machine. They define the specificity of people in terms of what computers cannot do. Simulated thinking may be thinking, but simulated love is never love. Women express this sentiment with particular urgency. We believe this is because a conflict fuels their conviction. A comfortable style of thinking would have them get close to the objects of thought. The computer offers them objects of thought. But the closer they get to this machine, the more anxious they feel. The more they become involved with the computer, the more they insist that it is only a neutral tool. A way out of the impasse would require profound change in the culture that surrounds the computer tool. If the computer is a tool, and of course it is, is it more like a hammer or more like a harpsichord?

The musician Robin is not distressed by her close relationship with her piano. A woman who finds attachment to the computer "unnatural" is not upset by her passion for the beautiful, heavy antique ink pens she uses to write. We infer that, if Lisa had been in music school, she would not experience as threatening her sense of communicating with her instrument or her emotional involvement with it. Music students live in a culture that, over time, has slowly grown a language and models for close relationships with music machines. The harpsichord, like the visual artists' pencils, brushes, and paints, is "just a tool." And yet we understand that artists' encounters with these can (and indeed, will most probably) be close, sensuous, and relational. And that artists will develop highly personal styles of working with them.

The development of a new computer culture would require more than technological progress

and more than environments where there is permission to work with highly personal approaches. It would require a new and softer construction of the technological, with a new set of intellectual and emotional values more like those we apply to harpsichords than hammers.[25] If computers are really the tools we use to write, to design, to play with ideas and shapes and images, they should not be addressed with the language of desktop calculators. Moving out of the impasse also would require the reconstruction of our cultural assumptions about hard logic as the ”law” of thought. Addressing this question brings us full circle to where we began, with the assertion that epistemological pluralism is a necessary condition for a more inclusive computer culture.

Roadblocks and openings for change

Achieving epistemological pluralism is no small thing. It requires calling into question, not simply computational practices, but dominant models of intellectual development and the rarely challenged assumption that rules and logic are the highest form of reason. Anne was able to escape the damaging effects of these models because, at her age, she was not expected even by the most committed Piagetian to have achieved the ”formal stage” -- and at the stage of ”concrete operations,” bricolage can seem acceptable. But in many educational settings, even this would not have saved her; her work with the computer would have been taken by teachers as the opportunity to encourage the development of ”more advanced” cognitive skills. She would have been given the message that her style of programming (and so, also, her style of thinking) was inadequate, that she was capable of something better, such as, ”the real thing.”

The message that she could do better would not necessarily lead someone like Anne to reject computers or ”do badly” with them on a technical level. We see many Annes following in the path of a Lisa or Robin. They respond to the dominant ethos of the computer culture by entering into an inauthentic relationship with the computer. This response can lead to a paradoxical reaction: frustrated bricoleurs appear at first sight to be extremely rigid ”planners.” Some turn to a ”cookbook” approach -- as when, in third grade, we were told to divide fractions by turning ”the second fraction” upside down. When denied a chance to do their ”real thinking,” they turn to rules that do not require them to think at all. People like Lisa and Robin ”escape to conformity,” a reaction that muffles the manifestation of a significantly ”different voice” in computing. But that voice is there. Recall the graduate student Lorraine, who says she tries ”to look like I’m doing what everyone else is doing” in order to preserve ”appearances.” Her style is hidden beneath her efforts to fit in.

We have said that, in our culture, the structured, plan-oriented, abstract thinkers don’t only share a style but constitute an epistemological elite. Language such as ”pure science” and ”pure mathematics” implies that their superiority is achieved by filtering out the concrete, and this means a continual put-down of people like Lisa and Robin. But although this way of thinking is deeply entrenched, there is cause for measured optimism. We conclude by describing how one opening for change is coming from within the technological culture itself. It takes the form of a new emphasis on computational objects, which is making itself felt in domains as diverse as deba-

tes about which personal computers are the best and how to build artificial brains.

Its simplest manifestation is the fashion for using icons in controlling personal computers. Consider two ways of getting a computer to copy information -- for example, the text of a section of this chapter -- from one diskette to another. In a traditional computer operating system, this requires typing an instruction. In an iconic system, the same effect may be achieved by moving a screen symbol for the text on top of a screen symbol for the diskette. The current technology for the act of moving something on a screen falls short of what the computer industry expects to provide quite soon, but existing systems, such as the ”mouse” or touch-sensitive screen, already give a tactile sense that recalls Anne's experience of programming as collage.

Even superficial use of icons is enough to transform the perception of the computer by people who are using it in computationally simple ways. For example, many writers who began to use computers reluctantly, as a necessary evil, are finding that warmer relationships are mediated by the icons, the mouse, and the cozier appearance of a Macintosh. And although these particular warmer relationships do not involve programming, their cultural influence means that the next generation people like Lisa and Robin will come to programming courses with a different sense of who ”owns” the computer.

But a multiplicity of technical methods doesn't by itself lead to pluralism. It can simply lead to competition. This point has been recently dramatized by the terms of the competition between the IBM PC and the Apple Macintosh computers. Many have been party to heated conversations between those who argue the superiority of each. (In the IBM the typical interaction with the computer is typing an instruction; in the Macintosh it is manipulating a screen object.) But when we understand the computer as a projective screen for different approaches to knowledge, we can listen to these conversations in a new way. Different people are comfortable with each system. When people fight about the IBM versus the Macintosh, what they are really trying to do is defend their cognitive style. And yet, the debate has both industry and consumers arguing in terms of whether IBM or Apple got it ”right.”

The Logo language allowed Anne and Alex to program in their own styles. But in many educational settings where Logo is defined as the computer language for children who have not reached the top stage in Piaget's hierarchy, allowing even as sophisticated a thinker as Anne or as creative a thinker as Alex to use their styles would be bought at the cost of defining their intellects as immature. Similarly, the very success of the Macintosh has often been cast in terms that reflect the elitism of the dominant computer culture. The Macintosh iconic interface has been brilliantly marketed as ”the computer for the rest of us,” with the implication that ”the rest of us” need things made simple and don't want to ”be bothered with technical things.” And from the beginning, the implication has been that it is a good system for women and children. If it is, it is not, in any simple sense, because it is ”easy.” When the Macintosh is experienced as good, it is experienced as good because, for some people, it feels like a thinking environment that ”fits,” while for others, and for very different reasons, the IBM feels like a thinking environment that ”fits.”

As it happens, the Macintosh's iconic style may be winning this argument. The designers of computer interfaces might interpret this as final proof of the technical superiority of icons. A

psychologist might read it as putting in question the hard/soft split. Perhaps everyone is really ”soft” after all, and ”hard” is a construct that is dropped when it is not needed for acceptability or prestige or functionality. Others might simply say that icons are ”easier.” All of the above may be in part true. But from our perspective what is important is that the iconic victories are part of a larger cultural shift towards an acceptance of concrete, relational ways of thinking.

The icons in the Macintosh reflect something deeper, a philosophy of ”object-oriented programming.” In the traditional concept of a program the unit of thought is an instruction to the computer to do something. In object-oriented programming the unit of thought is creating and modifying interactive agents within a program for which the natural metaphors are biological and social rather than algebraic. The elements of the program interact as would actors on a stage. This style of programming is not only more congenial to those who favor soft approaches, but puts an intellectual value on a way of thinking that is resonant with their own. It undermines the elitist position of the ”bards” in two ways. First, within the world of programming, it legitimates alternative methods. Second, in the larger intellectual culture, it supports trends in cognitive theory that challenge the traditional canon.

Until recently, prevailing models of cognitive theory have bolstered the commitment of psychologists and educators to the superiority of algorithmic and formal thinking. They were given support by the cognitive theorists most influential in the computer world, the leaders of the artificial intelligence community. In the late 1970s and early 1980s, the model of AI with the greatest visibility was the rule-based ”expert system” with its model of mind as a structured information processor. Critics of how computers influence the way we think cited the information-processing model as demonstrating the instrumental reason and the lack of ambiguity allegedly inherent in all computational thinking about intelligence (see, e.g., Dreyfus, 1979; Weizenbaum, 1976). But artificial intelligence is not a unitary enterprise. And recently, another model has become increasingly prominent: ”emergent AI.” [26]

Emergent AI does not suggest that the computer be given rules to follow but tries to set up a system of independent elements within a computer from whose interactions intelligence is expected to emerge. Its sustaining images are drawn, not from the logical, but from the biological. Families of neuron-like entities, societies of anthropomorphized subminds and sub-subminds, are in a simultaneous interaction whose goal is the generation of a fragment of mind. We noted that these models are sometimes theorized in notions of ”mind as society,” where negotiational processes are placed at the heart of all thinking. Those who espouse and support such models are more inclined to find bricolage acceptable than are classical Piagetians. What concerns us here is not which of these trends in AI is ”correct,” just as we aren't advocating a choice between the use of icons and the use of textual instructions in computer operating systems. What does concern us is that the new trends -- icons, object-oriented programming, actor languages, society of mind, emergent AI -- all create an intellectual climate in the computational world that undermines the idea that formal methods are the only methods.

Thus, recent technological developments in interfaces, programming philosophy, and artificial intelligence have created an opening for epistemological pluralism. There is the possibility for new

alliances between computation and the theorists as well as the practitioners of a science of the concrete. We began by presenting the notion of epistemological pluralism by reference to three streams of thought which, although different in many ways, converge in reasserting the importance of things in thinking. We close by noting the opportunity for their theoretical unification.

Louis Althusser wrote about psychoanalysis that the important breakthrough was not any particular statement about the mind, but the step of recognizing the unconscious as an object of study that defines a new theoretical enterprise (Althusser, 1964-1965). Psychology had considered the rational and the conscious as the quintessential mental activity; Sigmund Freud shifted the ground to the irrational and the unconscious. The unconscious was not only given recognition as an important "factor," but became an object of science in its own right. Similarly, we believe there is an opening for a break with ways of thinking that take the abstract as the quintessential activity of intelligence. We believe that the three intellectual movements we have noted -- feminism, ethnography of science, and computation -- are elements of a sea change that would not only recognize concrete thinking as important, but promote it to an object of science in its own right.

On a more down-to-earth level, there is every reason to think that revaluing the concrete will contribute to a computer culture that treats the computer as an expressive medium and encourages differentiated styles of use and relationship with it. There is every reason to think that this computer culture will be more welcoming and nurturing to women -- and to men. Gilligan has said that "women's place in man's life cycle" is to protect the recognition "of the continuing importance of attachment in human life" (Gilligan, 1982, p. 23). We conclude with an analogous point. The role of feminist studies in the nascent computer culture is to promote the recognition of diversity in how we think about and appropriate formal systems and encourage the acceptance of our profound human connection with our tools.

Footnotes

1. Edited collections include Bleir (1986) and Harding and Hintikka (1983). An overview that highlights many of the issues we deal with in this essay is provided by Elizabeth Fee (in Bleir, 1986). In this chapter we situate our position by focusing on two writers, Carol Gilligan and Evelyn Fox Keller. Gilligan, with her emphasis on moral discourse, might seem out of place in a discussion of noncanonical approaches to science and technology. But here we argue that key issues in the critique of science are not about scientific reasoning but about reasoning. Juxtaposing moral and computational reasoning helps us make this point. In addition, Gilligan's critical relationship to the theories of Lawrence Kohlberg is analogous to our own critical relationship to Piaget's work. We emphasize Keller because her work underscores, as does ours, the importance of relationships with objects in the development of noncanonical styles. Using Gilligan and Keller as a contrasting pair allows us to highlight two different dimensions of what we call the "soft" approach to science. See Gilligan (1982), Keller (1983, 1985).

2. A sample of relevant studies in scientific ethnography is provided by Knorr-Cetina and Mulkay (1983). See also Knorr-Cetina (1981), Latour and Woolgar (1979), Traweek (1989). A sample of studies on everyday thinking is contained in Rogoff and Lave (1984). Also see Lave (1988).

3. The Macintosh's replacement of proposition-like commands by the use of concrete icons has theoretical roots in a style of programming usually called "object oriented." For a nontechnical discussion, see Kay (1977, pp. 230-44); (1985, pp. 122ff.). The reaction within artificial intelligence against abstract, propositional, rule-driven methods was given literary expression in the writings of Douglas Hofstadter. See, for example, Hofstadter (1985, pp. 631-65). Two other manifestations of this reaction are Minsky (1987) and Rumelhart, McClelland, and the PDP Research Group (1986).

4. For a fuller discussion of the computer as an evocative and concretizing object see Turkle (1984).

5. Gilligan (1982). For a critical discussion of Gilligan's proposals and her reply, see Kerba, et al. (1986, pp. 304-33). Its methodological criticisms of Gilligan's treatment of the relationship between "voice" and gender do not detract from how her subjects illustrate the way of thinking we shall call "bricolage."

6. Research reports that emphasize approach to programming or programming style in the sense we are using it here include Papert, de Sessa, Weir, and Watt (1979), Turkle (1980, 1984, esp. chap. 3), Weir (1987), Turkle, Schön, Nielsen, Orsini, and Overmeer (1988), Motherwell (1988), and Harel (1988).

7. Lisa and Robin were part of a larger study of Harvard and MIT students taking introductory programming courses. The study found anxiety about an identity as a "computer person" to be an important aspect of reticence toward computers, especially among women. See Turkle (1988). See also Kiesler, Sproull, and Eccles (1985).

8. See, for example, the ethnographic studies referenced in Note 3 and the writings on scientific epistemology from the tradition of feminist scholarship referred to in Note 1.

9. Empirically, we sometimes find each aspect of soft mastery - bricolage as a style of organization and closeness to the object - without the presence of the other. In particular, one finds people who are planners but who enjoy a close relationship with concrete objects (and who experience computational objects this way). On the pairing of planning and what they call an interactive style with the computer, see Sutherland and Hoyles (1988).

10. In our research, the male/hard and female/soft dichotomy was most dramatic in a predominantly white, wealthy private school in the South, where traditional patterns of socialization would favor boys learning the ways of control, hierarchy, and distance, and girls learning the ways of negotiation and closeness.

11. Levi-Strauss (1968). Levi-Strauss contrasted bricolage with Western science, ignoring the significant aspects of bricolage present in the latter. Several recent writers have written in a way that begins to redress this imbalance. See, for example, Feyerabend (1975), Hanson (1958), and Wittgenstein (1953). In a less formal vein, see Feynman (1985).

12. In its ideal, the structured method would have the programmer go beyond subprocedures to make one procedure that could be given different parameters to produce arms and legs, right and left sides, even differently shaped people. This aesthetic, known as procedural abstraction, wants to see a right arm and a left leg disappear into a generalized abstract idea of "limb." But for someone like Alex, the top priority is staying in touch with the concrete. He is aware of the importance of organizing his program in order to find his way around it, but he does so by giving it what he calls "rhythm" rather than a hierarchical structure of procedures and subprocedures.

13. Although we have described Anne's program elsewhere, we redescribe it here in enough detail for the reader to appreciate how the concrete and the formal can come to the same place by alternative routes. Anne's program has the merit of showing in compact form a set of qualities characteristic of the bricoleur that are usually more diffusely represented. See Turkle (1984, pp. 110-115).

14. Bricolage does not exclude the use of subprocedures; it simply does not give their a priori delineation the status of a privileged method. Some ways that bricoleurs use subprocedures in a way that feels natural to them are captured in the following examples. First, a part of a program first conceived holistically can be demarcated as a subprocedure at any stage of programming. Second, subprocedures need not be "black boxes"; they too can grow by sculpting as the program grows as a whole. Finally, the bricoleur may use as subprocedures programs that happen to be "lying around," possibly even programs that were originally made for very different purposes.

15. Gilligan (1982). Kohlberg had already been challenged on other grounds. See, for example, Gibbs (1977). Similar issues have been raised in critiques of Jean Piaget. See, for example, Toulmin (1972). Toulmin argues that Piaget's experimental investigations reflect an a priori commitment to a Kantian position. We single out Toulmin because, unlike most of Piaget's critics, he does not quarrel with the detail of how the stages are described but with the epistemological assertion of the final end point.

16. Keller (1983, p. 198). Keller describes McClintock's approach as dependent on a capacity to "forget herself," immerse herself in observation, and "hear what the material has to say."

17. These experiments with Lego and programming are undertaken in a Piagetian spirit. See Piaget (1951) for experiments that deal with how mechanisms work. For a personal statement about the power of gears as an introduction to formal systems, see Papert (1980).

18. Our sample does not allow us to say that girls did systematically better than boys. Research is in progress on this point. Our present discussion is about styles of explanation (rule-driven vs. body syntonic), not distribution of abilities.

19. For example, most of those inspired by the Carnegie-Mellon schools of artificial intelligence. See Michalski et al. (1983).

20. Among the most influential writings that integrate this hypothesis are Keller (1985) and Chodorow (1978).

21. For an excellent overview of the object relations perspective, see Greenberg and Mitchell (1983).

22. In contrast, D.W. Winnicott has some suggestive ideas about the power of the "transitional object" - the baby's blanket, the teddy bear - that, in developmental terms, mediates between experience of self and nonself. In the current context, it suggests the power of the inanimate in inner life.

23. The Logo turtle was designed to be "body syntonic," i.e., to allow users to put themselves in its place. When children learn to program in Logo, they are encouraged to work out their programs by "playing turtle." The classic example of this is developing the Logo program for drawing a circle. This is difficult if you search for it by analytical means (you will need to find a different equation), but easy if you put yourself in the turtle's place and pace it out. (The turtle makes a circle by going forward a little and turning a little, going forward and tuning a little, etc.) Turtles are a path into mathematics for people whose surest route is through the body. See Papert (1980c).

24. See Turkle (in press). For more on women and the romantic reaction, see Turkle (1988a).

25. On values for a new computer culture, see Papert (1987).

26. For more extended comments on the "two AIs," see Papert (1988b).

References

Althusser, L. (1964-1965, December-January). Freud et Lacan. La Nouvelle Critique, Nos. 161-162.

Bleir, R. (Ed.). (1986). Feminist approaches to science. New York: Pergamon.

Chodorow, N. (1978). The reproduction of mothering: Psychoanalysis and the sociology of gender. Berkeley, CA: University of California Press.

Davis, P. J., & Hersh, R. (1981). The mathematical experience. Boston: Houghton Mifflin.

Dreyfus, H. (1979). What computers can't do: The limits of artificial intelligence (2nd ed.). New York: Harper and Row.

Feyerabend, P. (1975). Against method: The outline of an anarchistic theory of knowledge. London: NLB.

Feynman, R. (1985). Surely you must be joking Mr. Feynman. New York: Norton

Gibbs, J. (1977). Kohlberg's stages of moral judgement: A constructive critique. Harvard Education Review, 47(4), 43-61.

Gilligan, C. (1982). In a different voice: Psychological theory and women's development. Cambridge, MA: Harvard University Press.

Gilligan, C. (1988). Two moral orientations. In C. Gilligan, J. V. Ward, & J. M. Taylor (Eds.), Mapping the moral domain. Cambridge, MA: Harvard University Press.

Greenberg, J. R., & Mitchell, S. A. (1983). Object relations in psychoanalytic theory. Cambridge, MA: Harvard University Press.

Hanson, N. R. (1958). Patterns of discovery. Cambridge, UK: Cambridge University Press.

Haraway, D. (1979). The biological enterprise: Sex, mind, and profit from human engineering to sociobiology. Radical History Review, 20, 206-237

Harding, S., & Hintikka, M. B. (Eds.). (1983). Discovering reality: Feminist perspectives on epistemology, metaphysics, methodology, and philosophy of science. London: Reidel.

Harel, I. (1988). Software design for learning: Children's construction of meaning for fractions and Logo programming. Unpublished doctoral dissertation, MIT, Cambridge, MA.

Hofstadter, D. (1985). Waking up from the Boolean dream, or subcognition as computation. In D. Hofstadter (Ed.), Metatmagical themas: Questing for the essence of mind and pattern (pp. 631-665). New York: Basic Books.

Kay, A. (1977). Microelectronics and the personal computer. Scientific American, 237, 230-244.

Kay, A. (1985). Software's second act. Science, 85, 122.

Keller, E. F. (1983). A feeling for the organism: The life and work of Barbara McClintock. San Francisco: W. H. Freeman.

Keller, E. F. (1985). Reflections on gender and science. New Haven, CT: Yale University Press.

Kerba, L. K. (1986). On In a Different Voice: An interdisciplinary forum. Signs, 11(2), 304-333.

Kiesler, S., Sproull, L., & Eccles, J. S. (1985). Poolhalls, chips, and war games: Women in the culture of computing. Psychology of Women Quarterly, 9.

Knorr-Cetina, K. (1981). The manufacture of knowledge: An essay on the constructivist and contextual nature of science. Oxford: Pergamon.

Knorr-Cetina, K., & Mulkay, M. (Eds.). (1983). Science observed: Perspectives on the social studies of science. London: Sage Publications.

Latour, B., & Woolgar, S. (1979). Laboratory life: The social construction of scientific facts. Beverly Hills, CA: Sage Publications.

Lave, J. (1988). Cognition in practice: Mind, mathematics and culture in everyday life. Cambridge, UK: Cambridge University Press.

Levi-Strauss, C. (1968). The savage mind. Chicago: University of Chicago Press.

Merchant, C. (1980). The death of nature. New York: Harper and Row.

Michalski, R. S., Michalsky. J. G., & Mitchell, T. M. (Eds.). (1983). Machine learning: An artificial intelligence approach. Los Altos, CA: Morgan Kaufmann.

Minsky, M. (1987). Society of mind. New York: Simon and Schuster.

Motherwell, L. (1988). Gender and style differences in a Logo-based environment. Unpublished doctoral dissertation, MIT.

Papert, S. (1980a). The mathematical unconscious. In J. Wechsler (Ed.), Aesthetics and science. Cambridge, MA: MIT Press.

Papert, S. (1980b). The gears of my childhood. In S. Papert (Ed.), Mindstorms: Children, computers, and powerful ideas. New York: Basic Books.

Papert, S. (1980c). Mindstorms: Children, computers, and powerful ideas. New York: Basic Books.

Papert, S. (1987). Technological thinking versus computer criticism. Educational Researcher, 16(1), 22-30.

Papert, S. (1988). One AI or many. Daedalus. 117(1), 1-13.

Papert S., de Sessa, A., Weir, S., & Watt, D. (1979). Final Report of the Brookline Logo Project (Logo Memos 53 and 54). Cambridge, MA: MIT.

Piaget, J. (1951). La prise de conscience. Paris: Universitaires de France.

Piaget, J., & Inhelder, B. (1958). The growth of logical thinking from childhood to adolescence. New York: Basic Books.

Rogoff, B., & Lave, J. (Eds.). (1984). Everyday cognition: Its development in social context. Cambridge, MA: Harvard University Press.

Rumelhart, E. D., McClelland, J. J., & the PDP Research Group. (1986). Parallel distributed processing: Explorations in the microstructure of cognition. Cambridge, MA: MIT Press.

Sutherland, R., & Hoyles, C. (1988). Gender perspectives on Logo programming in the mathematics curriculum. In C. Hoyles (Eds.), Girls and computers (Bedford Way Papers, No. 34), London: Institute of Education, University of London.

Toulmin, S. (1972). Human understanding. Princeton, NJ: Princeton University Press.

Traweek, S. (1989). Beantimes and lifetimes: The world of high energy physicists. Cambridge, MA: Harvard University Press.

Turkle, S. (1980). Computer as Roschach. Society, 17, 15-22.

Turkle, S. (1984). The second self: Computers and the human spirit. New York: Simon and Schuster.

Turkle, S. (1988a). Computational reticence: Why women fear the intimate machine. In C. Kramarae (Ed.), Technology and women's voices: Keeping in touch. New York: Pergamon.

Turkle, S. (1988b). Artificial intelligence and psychoanalysis. Daedalus, 117(1), 241-268.

Turkle, S. (in press). Romantic reactions: Paradoxical responses to the computer presence. In M. Sosna & J. J. Sheehan (Eds.), Boundaries of humanity: Humans, animals, machines. Berkeley, CA: University of California Press.

Turkle, S., Schön, D., Nielsen, B., Orsini, M. S., & Overmeer, W. (1988).Project Athena at MIT. Unpublished manuscript.

Weir, S. (1987). Cultivating minds: A Logo casebook. New York: Harper and Row.

Weizenbaum. (1976). Computer power and human reason. San Francisco: W. H. Freeman.

Wittgenstein, L. (1953). Philosophical investigations. New York: Macmillan