Tracing the Dynabook: A Study of Technocultural Transformations

by
John W. Maxwell

MPub, Simon Fraser University, 1997
B.A. (honours), University of British Columbia, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate Studies
(Curriculum and Instruction)

UNIVERSITY OF BRITISH COLUMBIA

November, 2006

Abstract

The origins of the personal computer are found in an educational vision. Desktop computing and multimedia were not first conceived as tools for office workers or media professionals—they were prototyped as "personal dynamic media" for children. Alan Kay, then at Xerox' Palo Alto Research Center, saw in the emerging digital world the possibility of a communications revolution and argued that this revolution should be in the hands of children. Focusing on the development of the "Dynabook," Kay's research group established a wide-ranging conception of personal and educational computing, based on the ideal of a new systems literacy, of which computing is an integral part.

Kay's research led to two dominant computing paradigms: the graphical user interface for personal computers, and object-oriented programming. By contrast, Kay's educational vision has been largely forgotten, overwhelmed by the sheer volume of discourse on e-learning and the Web. However, an historical analysis of Kay's educational project and its many contributions reveals a conception of educational computing that is in many ways more compelling than anything we have today, as it is based on a solid foundation of educational theory, one that substantially anticipates and addresses some of the biggest civil/political issues of our time, those of the openness and ownership of cultural expression. The Dynabook is a candidate for what 21st-century literacy might look like in a liberal, individualist, decentralized, and democratic key.

This dissertation is a historical treatment of the Dynabook vision and its implementations in changing contexts over 35 years. It is an attempt to trace the development of a technocultural artifact: the Dynabook, itself partly an idealized vision and partly a series of actual technologies. It is thus a work of cultural history. But it is more than simply a looking back; the effective-history of the Dynabook, its various incarnations, and its continuing re-emergence and re-articulation mean that the relevance of this story is an ongoing question which needs to be recognized and addressed by educators, technologists, and learners today. This dissertation represents an introduction to this case.

Table of Contents

List of Figures

Acknowledgements

A good many people contributed in a variety of ways to my being able to take this project on and to complete it. My sincere and lasting thanks go to them.

To Ricki Goldman for making my path clear; Mary Bryson for her insistence on critical vision, and for her unshakable support; Gaalen Erickson for kindness, wisdom, and unflagging interest.

To Prescott Klassen for the sense of history and the encouragement to think big; David Porter for giving me creative opportunity and encouragement for my explorations; Rowly Lorimer for the space to think, to work, and to engage; Martin L'Heureux for reliable intellectual foiling, and for sharing much of the path; Avi Bryant for being an exemplar.

To Ward Cunningham for Wikis, especially c2 and for the incredible repository of software wisdom collected there; Simon Michael for ZWiki, which served as my writing environment; Wikipedia for more breadth than I have any right to.

To Pavel Curtis and Amy Bruckman for objects; Matthias Müller-Prove for blazing a path through the literature.

To Kim Rose for the direct line to the sources and for unfailing paitience; Ted Kaehler and Ann Marion for generosity of time and help far beyond what I asked for; BJ Allen-Conn, Bobby Blatt, and Dan Ingalls for their help in putting the pieces together; John Dougan for his help with the images.

To Alan Kay for a world to grow up in, and my Dad for seeing it.

To my wife, Kelly, for constant intellectual stimulation, for historiography, for patience, for superb copyediting, and for a home for me and my ideas; Norm for seeing what I was on about before I even did.

To my Mom for the solidest of foundations. To James and Ayla for the future.

Dedication

For James, Ayla, Anja, Cara, Maia, Adrienne, Julien, Ben, Joelle, Bram, and Natalia Joy.

Introduction

This is a story about educational computing—that is, computers in education. What does that *mean*, exactly? How we come to an answer to that question is a good deal of what the next 200 pages are about. That is to say, the question of what "computers in education" means isn't a simple one. It is not the kind of question we can answer in a sentence or two and then get on to the business of plugging in cords and training users. Rather, the meaning of computers in education is something that is *contested*, and has been contested for about forty years already. Over that time, the answer to the question has not become particularly clearer; on the contrary; I will take pains to argue here that it has looked substantially clearer at points in the relatively distant past than it does now.

The story I am about to tell is that of Alan C. Kay and his work on a possible vision of personal and educational computing; a research program which began in the early 1970s at Xerox' research labs in Palo Alto, California and which has gone through a variety of institutional contexts since then, even continuing today. Alan Kay's work is romantically described in a vision he articulated some thirty-five years ago under the rubric of the *Dynabook*, which continues today to act as a sort of touchstone and reference point for the ongoing development and evolution of a particular rendering of what personal and educational computing might mean.

Kay's story isn't well known, compared, for instance, with the story of Steve Jobs and Steve Wozniak inventing the Apple computer in their garage in the late 1970s, or of Bill Gates' founding of Microsoft corporation in that same decade. But despite its relative obscurity, I will argue that Alan Kay's story is one of the root texts in the construction of personal and educational computing. In delving into this history, and in evaluating our contemporary aporias in the light of it, I will argue that the *cultural trajectory* of personal and educational computing can be made better sense of—and that opportunities for personal agency, critical understanding, and political action appear—in the light of such a historical study.

A starting point for this research is the *constructedness* of personal and educational computing. Now, the constructedness of the world is a popular topic in recent social and cultural theory, but what is often missed is the element of ongoing political situatedness and the active and generative space opened up by critical engagement with these constructions. The world is not *given*, but is in large part created by participants in particular social, cultural, historic, and practical contexts. Moreover, the constructedness of the world does not have to be something over/against regular folks. A good part of my agenda here is to show that the construction of personal and educational computing is not something *done to* users, learners, teachers, or even critics. Personal computing is not given; it has been constructed through particular historical contingencies, and, more important, it is continually and ongoingly constructed. *What is a computer? What good is it? What is it for? How does it fit into our lives?* It is important to remember that these questions remain open. The work of defining this ground is still going on; the game is still up for grabs.

I hope to show how Alan Kay's work—beginning in the 1970s—was the first major, sustained entrée into this realm: that is, the work of opening up the possibility of personal agency in the struggle for the construction of meaning and effective action in personal and educational computing. The considerable influence in this area weilded by large market-driven corporations like Apple Computer and Microsoft Corporation is altogether more recent. Furthermore, despite the apparent dominance of corporate market logic in defining the meaning and significance of personal and educational computing, I intend to show how attention to the history of this field can reveal opportunity for individual 'users'—however circumscribed their agency may appear in the face of corporate domination or the threatening chaos of the Internet.

The apparent disempowerment of individual learners, teachers, and other front-line 'users' in the face of a rapidly growing and complexifying world of computing and digital media is the target of this work. In my studies of educational computing, I am repeatedly faced with the challenge of making sense of a field which basically *does not make sense*—that is, it is without a guiding rationale or set of common principles which might guide action or

even critique. Educational computing seems a multi-headed and often self-contradictory beast, almost wilfully ignorant of its own history, and as a result often at the mercy of whatever fashions or—in the post-9/11 world—terrors may carry the day. The result is that whether the current obsession is to be upgrading our software, updating our blogs, or fending off network-borne viruses, the extent of most users' understanding and feelings of control over what they are doing is, to say the least, compromised.

The Story So Far—A Conventional Mythology

You may ask yourself, how did we get here? How do we find ourselves in a world dominated by an often overwhelming technological infrastructure, in which fear and insecurity have become such driving forces? In order to answer this question, we can begin by examining the conventionial history of personal computing, that which serves as the origin myth and working model of this world.

In the beginning—so the popular story goes—there were mainframes; computers were enormous, air-conditioned beasts tended by teams of white-coated priests (or, alternatively, by teams of post-WAC gals carrying reels of tape and patch cables). In these early days, the story goes, computers were put to large, institutionalized purposes: taxes, billing, artificial intelligence, and world domination: somehow these increasingly large and powerful machines would surely break their chains and devour humanity, or at least enslave us.

But the *promethean* leap apparently came in the late 1970s, when a ragtag army of hobbyists in southern California—working in pairs in garages—invented the personal computer out of spare parts and baling wire. These early computers were tiny and inexpensive, tended by greasy adolescents in dirty t-shirts. It wasn't long, however, before the smell of money mingled with the the odor of solder and the whiff of burning components. A cadre of early computer entrepreneurs—Steve Jobs, Bill Gates, et al.—set up shop to battle the established computer industry (IBM), who peered down from its mainframes and wondered what to do now that the secrets were out.

This 'new world' of computers is romantically captured in a new genre of computer magazines—like *BYTE*—that appeared in the late '70s and early '80s: half inch-thick glossy publications stuffed full of opinionated editorials, recipes for homebrew computer projects, and full-page colour advertisements for the new software "titles" which were appearing to tap this new market. Software appeared in three major categories: *office productivity software* like spreadsheets and word processors—the personal computer had terrible, unresolved issues with legitimacy and desperately desired to be accepted by real businesspeople; *computer games*—probably the most lucrative market of the three; and *educational software*—which often looked quite a bit like the games, for marketing reasons at least. Educational software tended to be drill-and-practice exercises, tarted up with as much colour and sound as the makers (and the existing hardware) would allow. And there was also *Logo*, an educational programming language developed for children and undoubtedly good for developing young minds, though it seemed no one was quite sure how. In any case, regardless of the intellectual depth (or lack thereof) of these works, it was enough to establish educational software as a persistent genre in the minds of the computer-buying public: one of the things that these machines were "good for," a reason for buying—or at least for justifying the purchase.

According to the popular story, three key events in the early 1980s rescued personal computing from its greasy hobbyist image (with a vaguely countercultural air about it) and made it into an economic powerhouse. The first was IBM's hugely successful introduction of the "IBM PC," which set the paradigm for what a personal computer was, almost completely eliminating all others (other players in the PC market made "PC clones" after this). IBM brought the respectability of established business, and poured marketing money into making the "PC" an indispensable part of small business operations.

The second—much more in keeping with the promethean mythology of personal computing—was Apple Computer's 1984 introduction of the Macintosh, branded "the computer for the rest of us." Apple's early marketing of the Mac lives on in popular history. With it, they simultaneously defined a legitimate alternative market for personal comput-

ers—easily reduced to "creative" types—and cemented IBM's mainstream business market. In caricaturing IBM's "big brother" image, Apple undoubtedly helped reinforce IBM as the market leader for serious computing. The paradigm—or genre—that IBM established was shored up by Apple's circumscription of the margins. This division lives on in the popular imagination to this day.

The third event was not so much a single event as a clear trend: video games' growth into an enormous, lucrative market. Atari was the market leader here in the early 1980s, dabbling in sales of both personal computers and dedicated video-games machines, but more importantly with the design and distribution of the games themselves, to whatever platform. If there was a question—*what are they for?*—surrounding personal computers, there was no such worry about video games.

Regardless of the ambiguity of what personal computers might be good for, the market and industry surrounding it grew with phenomenal energy through the 1980s and into the 1990s; a new breed of computer millionaire emerged in Silicon Valley, around Boston's "Route 128," and in several other centers in North America (notably Seattle, in Microsoft's case). There was money to be made, and between the innovating potential of digital technology and the gradually growing demand for it in the marketplace, personal computing flourished. More and more businesses, schools, and individuals bought personal computers; the industry steamed ahead with new and innovative uses for them: productivity software, educational software, games, and now interactive multimedia, graphics, audio and video tools. The "Multimedia PC" of the early 1990s, centered around its CD-ROM drive, pushed the market ahead again, and the growth of a content-based CD publishing industry seemed certain.

The key innovation to emerge in the 1990s, of course, was the World-Wide Web, which first reached public consciousness in 1994 and 1995. Almost overnight, the personal computer's identity shifted from that of productivity tool to information appliance, tapping a world-wide ocean of information; pundits waxed rhapsodic. For educational and personal users (that is, apart from the established office productivity market), the "Web" became the

single most important reason to own a computer, and the Web browsing software *Netscape Navigator* was proclaimed the new "killer app." Netscape Communications Co. raised 1.2 *billion* dollars in its now-famous Initial Public Offering (IPO) in 1995, sparking a flood of investment reminiscent of the California gold rush of 1849, or at least the Dutch tulip market of the 1630s. Through the late 1990s this new gold rush ran wild, with billions of dollars invested in driving innovation online. When the "tech bubble" finally began to subside (it would be an overstatement to say it burst) in 1999 it left in its wake a landscape littered with new technologies; some useful, many not, some significant, many soon forgotten. What had become clear was that the paradigm of personal computing had been firmly established throughout western society: a 2005 report, for instance, states that 75% of Canadians have a computer at home; 72% are Internet users. More than 30% get their daily news online (Canadian Internet Project 2005). A 2004 Statistics Canada report states that 97% of Canadian schools were Internet connected, with an average of 5.5 students per connected computer (Statistics Canada 2004).

One more important statistic is this one: the number of people online—that is, capable of communicating on the Internet—is *one billion*, as of late 2005, according to Mary Meeker of Morgan Stanley Research.[1] A billion people makes for a large-scale, complex society by any measure. And yet, our primary means for interacting with this complex environment is the personal computer, a bastard, haywired-together technology born a scant two-and-a-half decades ago by greasy youths in garages in California, sold mostly by consumer-electronics hucksters in the intervening years, and developed largely via gold-rush hysteria. What we've inherited is the *PC as generalized interface* to a big, scary world out there. But it is significantly underpowered in comparison to the task; I do not mean here that the processing power, the MHz, or the RAM is insufficient—what I mean is that what has become a significant communications medium—a major established genre or paradigm of human expression, communication, and commerce—is built on extremely shaky founda-

1. Interestingly the study reports that 36% of those users are located in the Asia-Pacific region; while only 23% are in North America. See Meeker (2005).

tions, and patched up and reinforced over the years with little more than glossy magazine advertisements. A hundred years ago, the exigencies of the book publishing world led printers increasingly to use cheap pulp paper, despite the fact that pulp paper disintegrates into dust within about a century under most conditions. But this is vastly more robust than the state of the personal computer, which threatens to burst asunder for many "users" on almost a daily basis, in the face of quotidian bugs, virulent viruses, overwhelming spam, software piracy, invasion of privacy, pop-up pornography, chat-room pedophilia, and general information overload.

Now, fear and loathing have never been serious impediments to commerce or progress; indeed, they are often powerful drivers. The personal computing market is certainly driven by such forces, and educational computing is no different. Far from "personal" computer users—a collective which, at numbers like those quoted above, is roughly equivalent to "citizens"—being in any kind of control of the digital world, the real battle to control the discourse is fought by large and mighty corporations. Microsoft, for one (and they are certainly not alone in this), has established itself as an immense, indispensable part of the environment by offering to manage the interface between 'users' and the vast, ambiguous, frightening, and complex world of technology and the Internet. That they have been accused on many occasions of being more part of the problem than the solution matters little; Microsoft's marketing genius—a paradigm-defining one—is in understanding and managing just how much or how little consumers want to know, or understand, about what goes on beyond their monitor screens. It is not a stretch to say that all successful technology companies today succeed because they play this particular game well; consider Google's enormously successful management of online content (and the dearth of attendant critique). In education, WebCT, one of the most influential companies in educational technology today, succeeds precisely because of their successful control of the ambiguities and complexities of the environment in which their customers need to work. This is the dominant dynamic of the first decade of the 21st century.

A Critique of the Popular Mythology

Such is the conventional story of personal computing. This is the mythology of this moment in time, the history which makes sense of the world we live in. It is, of course, only one story, and it is inadequate and indeed obfuscating on several levels.

It is helpful to look at the story of personal computing as one emergent within a context of contemporary journalism, advertising, and marketing, for these are the main arenas in which the conventional story has played itself out so far. To the extent that popular journalism and advertising constitute public discourse, this is in fact and practice *our story*. But it is not difficult to problematize this. A simple tactic is to simply look for what is absent.

In the first place, there is practically nothing about "computer science" in the story; it plays out as though the formal, academic study of computing (half a century old) did not exist, or perhaps as if this realm were some dusty, antiquated pursuit that we were better to have left behind in the promethean moment of the late 1970s.

The second major absence is that of software. The conventional story, as reported and advertised in newspapers and magazines, and played out in catalogues and showrooms is overwhelmingly concerned with computer hardware. Software, when it is considered at all, remains in its standard-sized, shrinkwrapped boxes. Personal computing has largely been about personal computers, as artifacts, commodities, toys, gadgets. There is very little about what actually goes on inside these computers, even in the face of the obvious and oft-repeated fact that the wealthiest man in the world, Bill Gates, headed a company that doesn't deal in hardware at all. Somehow, the fetish is entirely physical, and we have come to accept that software is a necessary evil that allows the hardware to work, and which somehow slouches toward its slow improvement. Presumably, it is easier to talk and write about hardware than software. The finer points of chip design are buried deep within the black box—or rather, the shiny exterior (or at least the beige plastic cases) of the machine; the details of software are actually in our faces more than we like to admit, but besides a few trite discourses (GUIs vs command line; Mac OS vs Windows), this fails to get the attention that

hardware does. When CD-ROMs appeared in the early 1990s, and afterward the Internet, we began to talk about "content" with respect to computers, despite the fact that we rarely speak of digital content in ways that are any different from the content that appears in books or on television. But our conception of "content" is nowhere near sufficient to grasp the significance of software today.

The third conspicuous absence in the conventional story is history itself. The sheer volume of discarded computer hardware suggests an alarming tale which appears now and then amid reports of sending old PCs to Africa, like eyeglasses in the Second Sight project. But nothing is ever said of the volume of discarded effort spent designing, developing, learning, and using the software of years past. With the exception of a persistent genre of old-timers' reminiscing their old beloved version of Word (or Wordperfect, of Star Writer, or whatever—always writers talking about word processors) long past, we give close to zero thought to the decades of evolution of software. The mythology seems to prescribe that the newer is always a straightforward improvement on the older (usually along the lines of *more better faster cheaper*), and wholesale innovations (the web browser, for instance) accepted as being born fully formed from the foreheads of their developers. This obsession with the march of the new masks not only the person-years of toil and thought, but also the myriad missed steps and missteps along the way. It masks, fundamentally, the constructivist's cry, "It could have been otherwise."

Impoverished perspectives

The conventional story of personal computing is caught between the twin horns of two popular caricatures of technology: instrumentalism and determinism.

Instrumentalism is the simple and common belief that we create technologies to achieve particular ends, to solve particular problems. The assumption in instrumentalism is that these ends or problems are clearly defined in advance, such that technological solutions can straightforwardly be specified and developed. Instrumentalism further carries with it

the assumption that technology is value-neutral, a mere tool in the hands of a purposeful designer or user.

Technological determinism is in some ways the mirror-image of instrumentalism; the determinist perspective holds that technology has a logic of its own: most fundamentally, that progress is inevitable, towards better and better ends (this the Enlightenment's position) or toward more sinister and oppressive ends (the position of much critical theory and a good deal of latter-day science fiction).

It is easy to pose these two stances against one another, and view the world of technology as a struggle between the two or as a playing-out of a middle ground or compromise. I think it better to see instrumentalism and determinism as commonplace perceptual facets of technological systems, which appear 'naturally' to us in differing circumstances, but which fail in most cases to really focus our attention or provide a useful analytical framework: we look at advertisements for new cell phones that can record movies and download coupons and we muse, "what next?" in a happily determinist frame of mind. We purchase the next iteration of the cheap disposable inkjet printer in a spendthifty instrumentalist mode. And then we wade through mountains of spam in our e-mail in-boxes and curse that the Internet is out of control.

What to do? And how could we know anyway, given that our thinking about technology is so circumscribed? We need to remember—despite the constant temptation not to—that how we confront problems and issues today is historically conditioned; we got to this point by way of a specific unfolding of circumstance. But historical awareness is limited; things haven't always been as they are, and they might have been otherwise, but it certainly does not follow that we can simply choose otherwise: to consciously adopt a different position.

Technology is political. It is not a neutral, external realm of human activity separate from political and ethical concerns. Neither is it an 'influence' on the ethical and political, nor are these facets of our lives mere 'influences' on technology. Rather, technology *is* politics and ethics—beginning right with our difficulty in remembering so. This is a stance which I will elaborate in some detail in the pages that follow. In particular, I want to spot-

light this notion with particular attention to computer software, a subset of technology which is more and more shot through our private and public lives. Software has always been political, but today, in the early 21st century, the politics of software have become acute. And while there is an emerging discourse and literature addressing this (e.g., see Lessig 1999; 2002b; Moglen 2000; 2003; Stallman 2001; 2003), it has not reached widespread public attention. I see this as a crisis facing Western societies (and by extension, everybody else, given the agendas of globalization). The reason for the lack of focus on the politics of software, despite the technological messes that accumulate around us, has to do with the basic ahistoricity in our thinking about technology. My method here is to lead with historicity, so that this moment in time can be framed, and so that the idea of software *as politics* has some concrete meaning.

The Division of Labour in Modern Technology

Let us begin with a particular question about technology, computers, and software: whose problem is this, anyway? Alternatively, we can ask: who's responsible for this mess?

The common and superficial response, which often bills itself as the humanist perspective, is that the designers and marketers of computing technology are responsible for the technological systems surrounding us. This argument casts our technological dysfunction in either a technological determinist light (Menzies 1989; Bowers 2000) or an instrumentalist one with a determined overclass: the military-industrial complex (Edwards 1996). While these treatments both correctly identify a nastily asymmetrical power dynamic surrounding technology, they run into trouble when they attempt to isolate the problem as external to the lifeworld of ordinary people—that technology is a system put over against 'us.' The characterization of computer technology as having been imposed upon society by an engineer/capitalist elite neatly divides up the responsibility for our ills: someone (industry, salesmen, zealous technologists, etc.) is to blame, and the analysis ends there. The resulting responses tend to impotence: whether we should enact laws (limiting corporate power; protecting individual privacy; protecting consumer's rights; regulating the Internet; etc.), or

'resist' technology (don't carry a cellphone; chop up your credit card; refuse to upgrade your word processor; computers out of the classroom), or write critiques and stern warnings about the fate of the world. These are all commonplace ideas; we all engage in many of these tactics—I certainly do.

There is an underlying and foundational trope lurking herein, though, and it hamstrings everything we might like to do about our technological predicament. The assumption is, broadly framed, that technology is an external force on our lives, driven by someone else's agenda. More specifically put, the assumption is of a division in society: a division of labour between *experts* and *end-users* (or producers and consumers). We willingly and unproblematically learn this division, choose it, take it on, and reproduce it. We reify it in our buying habits, in our curriculum plans, in our legislation, in our discourses. I would not claim that these power imbalances aren't very real, but we are doomed to live by their terms when we take on the roles assigned to us. But, of course, we're also stuck with them, and changing the world is not just a matter of changing one's shirt.

Now, it is not my intent to go into a lengthy discussion of hegemony or domination here. My purpose is rather to do the history of how we got to this particular place. In the hermeneutics of the historical process are—I optimistically believe—the generative possibilities. What can we know about the division of labour in information technology, between *experts* and *end-users*?

C.P. Snow's famous "two cultures" of the sciences and the humanities only begins to frame the division as it presents itself here; the computer age brings with it an economic and political apparatus that institutionalizes the producer/consumer divide on top of the expert/end-user division. The tension between expert knowledge and public dialogue is age-old. Latour identifies the origins of it with Socrates in Plato's *Gorgias,* in which politics is (mis)represented as one of right vs. might (Latour 1999, p. 219ff). Latour uses this as an analogy for our popular conception of the relationship of science to politics. Instead of calling for a science free of political influences, Latour wants a "politics freed from science"—that is, freed from the kind of political shortcutting it is often called upon to do: "a *substitute*

for public discussion" (p. 258). Long has "Science" (Latour uses the capital "S" in this rhetorical characterization) been called upon to end the messiness of actual political discussion: the introduction of the "impersonal laws" of nature as an antidote to the irrationalism and ambiguity of human judgement, and thus opposed to Politics as such. Latour presents an alternative "science" (without the capital) which involves the proliferation and extension of complex collectives of reason, argumentation, and agency which *are* political discourse. Latour's capital-S Science (or Reason) is thus a conventional tool for silencing one's opponents, but he reminds us that this version of science is not the whole story, and that there is no analytically convenient "inside" and "outside" of science (1987, p. 145ff). Latour is concerned too with the division of labour.

Complicating this account, however, is the work of technology theorist Arnold Pacey, who wrote on the "culture of expertise:" Pacey offers an argument for a specially situated kind of technological determinism or "technological imperative" at work within groups of engineers and technologists, that an airplane such as the French/British *Concorde* would never have emerged apart from a drive for engineering excellence in itself. Pacey cites Freeman Dyson on nuclear weapons: that their existence is in part due to the *telos* of the culture of expertise, that they are "technically sweet" projects that appeal to physicists, as opposed to the hard, hacked out engineering of conventional weapons (Pacey 1983, p. 43). What is amiss in this kind of a world, Pacey suggests, is the compartmentalization of our values within various spheres of activity (public, private, men's, women's, educational, professional, etc.), and that a solution might be a broad-based effort to break down these compartmentalized traditions and virtues.

What this means to me is that the divide between inside and outside, or between expert and everyman, is not one that can merely be undone or unbelieved in; rather, it is a cultural phenomenon that we are dealt. Latour's two characterizations of S/science are, historically speaking, both actual, and actively in tension. Pacey's observations point to the fact that we continue to reify the poles of the division. The more we believe in them, the more real they become. The longer we believe in *end-users*, the more distant we become from the *expert*

pole. The result is a disastrous commonplace sensibility about technology's place in society. Within education, computational-literacy advocate Andrea diSessa described what he calls the "culture gap," characterized by an "anti-learning bias" on the part of technologists, an insistence on superficial transparency of computing artifacts, and a deep-seated expectation that only some individuals can assume (professional) positions of knowledge and authority—a notion which brings with it distrust of broad-based competence (diSessa 2000, p. 225ff, 237). This is but one particular articulation.

Similarly, much of the literature on computing that falls roughly within the Science, Technology, and Society (STS) rubric (e.g., Turkle 1995 and others in Shields' 1995 volume) is unfortunately inscribed in the stereotypical humanist vs engineer division. The result is analysis that says 'the engineers only thought of things from the engineering perspective, and have imposed solutions on us that fail to take into consideration what we humanists need.' While there is undoubtedly a grain of truth expressed in this, it is but vanity to construct this as a story of oppression from above. To make it into such a moral position does considerable violence to the discourses contextualizing the engineers' work—as if they were working in isolation. Turkle's 1995 analysis is a case in point: she reports on the MIT's Project Athena, an enormous effort to computerize an entire campus in the 1980s. Turkle cites Project Athena's ban on the progamming language BASIC, followed by a reversal under considerable pressure, but with the condition that BASIC would remain officially unsupported. Her account points this out as an example of the arrogance of the systems and administrative people in the face of 'real world' needs of users. The critique, however, is predicated on the assumption of an insider/outside split: engineers vs. humanists; developers vs. end-users; experts vs. regular folks.

But such divisions, no matter how commonplace or self-evident they may appear (reification works thusly), are *caricatures*; they fold into non-existence the untold hours of *labour* that go into the design and maintainance of systems, the extensive and complex networks of discourse and practice that must be created and sustained in order for such

systems to ever exist, and the deep points of connection that actually bind together the people and machines and systems on both sides of the apparent divide.

There are, luckily, alternative conceptions. Two of the strongest, from the science studies literature, and which serve as touchstones for me, are the writings of Bruno Latour and Donna Haraway. Both thinkers are bloodhounds on the trail of taken-for-granted boundaries. Latour's boundaries are those that separate science from politics, society from nature, human from nonhuman; these are mythological artifacts of the 'modern' age, Latour argues (1987; 1993). Latour counters that there is no inside or outside of science, only a proliferation of hybrids. Donna Haraway's target boundaries similarly are those which purportedly guarantee purity of a privileged conception of humanity. Her powerful contention (1991) is that the cyborg is us, we are always already compromised and impure, hybrids political and natural, material and semiotic, technical and moral.

The situated stance taken by Haraway is significant: we are not in a position to re-invent the world wholesale, but rather to fight for a fairer distribution of power, one not so overwhelmingly dominated by entrenched institutional power bases. This is not an all-or-nothing struggle, but rather a tactical strategy to spread the fruits of technoculture around more evenly. Technology, especially in its digital form, need not only be the instrument of established power to maintain and extend itself. That's what I mean by technology—and especially software—being political: it is an active politics that works bidirectionally; it is generative, as the Foucauldians have pointed out. There is actually a strong tradition of this sort of work and thinking in computing, in the academy, in education. And indeed, it is my contention that Alan Kay's body of work of speaks very clearly to this issue.

How Education is Complicit

The conceptual rift between 'experts' and 'end-users' is thriving in our educational institutions. The whole field of educational technology is based on a confused discourse about ends and means; it reifies experts and end-users, technological means and pedagogical ends, as if these were pre-existing categories. And in a sense they are, as the academic world is

similarly predicated on this division of labour: researcher vs. researched, subject vs. object. The technological aspect then is symptomatic of a larger unquestioned division between experts and non-experts, making it a structural or systemic issue. The sheer volume of history—of tradition and culture—underlying this division of labour issue is immense: it goes right to the core of modernism and capitalism and science and our very way of being in the world. It has everything to do with how we inscribe the boundaries of technoscience—the structure of the economy, our construction of gender and class, our expectations about freedom and choice, our acquiescence and resistance to globalization and corporatization, our expectations about public vs. private vs. common.

Within educational technology, the division of labour manifests itself along a number of different axes. In the first and most obvious case, educational institutions' uncritical acceptance of industry-originated 'solutions' and large-scale buy-in to marketing campaigns contribute substantially to the establishment of the subject positions which disempower pretty much everybody involved: students, teachers, and the schools themselves. I will not go into this at length here, as the general phenomenon of the corporatization of schools has been dealt with elsewhere (e.g., Bromley & Apple's 1998 volume, *Education/Technology/Power*). The superficial appeal of industry-based solutions is easy enough to see: the difficult initial design and implementation work are taken on by an industry 'partner,' thereby freeing the school or college to concentrate on their core business: education. Of course, what's missing from this particular division of labour is any developed sense that *the one may have an impact on the other*: the 'problem' to which the 'solution' is an answer is one pre-defined by the vendor. A recent example is Apple Computer's offering sets of wireless laptops to educational institutions; it is not at all clear what problem this solution actually addresses. The superficial answer was that learners would be freed from computer labs, but Apple's wireless-laptop scheme looked remarkably like computer labs on wheels: access to machines still had to be booked, hardware locked down to prevent theft, and, most importantly, the machines were still (ironically) 'time-shared,' as computer labs have been for thirty or forty years.

A second manifestation of the expert/end-user divide is perhaps best articulated with reference to the "miracle-worker" discourse:

> This apparent and longstanding lack of success in reaching implementation goals with respect to uses of digital tools in schools has created a specific niche for the working of miracles—the provision of digitally mediated environments within which to re-mediate the production of knowledge in educational contexts...
>
> Within such a context, the miracle worker's effectiveness is measured by their capacity to spin narratives of success against all odds by providing tools, but more often discourses, that appear to transform students' engagements with information. (de Castell, Bryson, & Jenson 2002)

The "miracle worker" discourse reinforces the machinery of desire that is central to the marketing efforts of high-tech vendors. Seen on this level, that the individual 'miracle worker' is predictably non-duplicatable—or at least 'unscalable'—is unfortunately almost *the point*. While we love to love those who take the initiative to make a real difference in their schools and who personally drive innovation, the too-common reality is that when these few individuals burn out, retire, or take advantage of their technical expertise and get a higher-paying job, what is left is a reminder of how wide the gap really is, setting the stage for the next round of marketing campaigns.

In a third manifestation, the trend toward online distance education, "distributed learning," "learning objects," and so forth establishes an even more cynical (or at least 'closed') position, quite comparable to the textbook publisher, in which all knowledge and authority is vested with the publisher/information source and the model is a simple instructionist one of transferring this information to the user. As with the solution-provider discourses, the information-provider discourse makes plenty of sense in terms of business models, but not so much for learning. The "distance-ed" variety of this discourse is the centralized version, while the "learning objects" version is a distributed market economy; either way, the educational process is one way, and reliant on an 'impoverished' recipient.

A fourth manifestation of the expert/end-user divide within the educational environment may be more damaging than any of the above: in this case, the critical faculties of the educational establishment, which we might at least hope to have some agency in the face of large-scale corporate movement, tend to actually disengage with the critical questions (e.g., what are we trying to do here?) and retreat to a reactionary 'humanist' stance in which a shallow Luddism becomes a point of pride. Enter the twin bogeymen of instrumentalism and technological determinism: the instrumentalist critique runs along the lines of "the technology must be in the service of the educational objectives and not the other way around." The determinist critique, in turn, says, 'the use of computers encourages a mechanistic way of thinking that is a danger to natural/human/traditional ways of life' (for variations, see, Davy 1985; Sloan 1985; Oppenheimer 1997; Bowers 2000).

Missing from either version of this critique is any idea that digital information technology might present something worth actually engaging with. De Castell, Bryson & Jenson write:

> Like an endlessly rehearsed mantra, we hear that what is essential for the implementation and integration of technology in the classroom is that teachers should become "comfortable" using it. [...] We have a master code capable of utilizing in one platform what have for the entire history of our species thus far been irreducibly different kinds of things—writing and speech, images and sound—every conceivable form of information can now be combined with every other kind to create a different form of communication, and what we seek is comfort and familiarity? (2002)

Surely the power of education is *transformation*. And yet, given a potentially transformative situation, we seek to constrain the process, managerially, structurally, pedagogically, and philosophically, so that no transformation is possible. To be sure, this makes marketing so much easier. And so we preserve the divide between 'expert' and 'end-user;' for the 'end-user' is profoundly she who is unchanged, uninitiated, unempowered.

The result is well documented: scores of studies show how educational technology has no measurable effect on student performance. The best articulation of this is surely Larry

Cuban's (1986) narration of the repeated flirtation with educational technology; the best one-liner the title of his article, "Computers Meet Classroom: Classroom Wins" (1993). What is left is best described as *aporia*. Our efforts to describe an instrumental approach to educational technology leave us with nothing of substance. A seemingly endless literature describes study after study, project after project, trying to identify what really 'works' or what the critical intercepts are or what the necessary combination of ingredients might be (support, training, mentoring, instructional design, and so on); what remains is at least as strong a body of literature which suggests that this is all a waste of time.

But what is really at issue is not implementation or training or support or any of the myriad factors arising in discussions of why computers in schools don't amount to much. What is really *wrong* with computers in education is that for the most part, we lack any clear sense of what to do with them, or what they might be good for. This may seem like an extreme claim, given the amount of energy and time expended, but the record to date seems to support it. If all we had are empirical studies that report on success rates and student performance, we would all be compelled to throw the computers out the window and get on with other things.

But clearly, it would be inane to try to claim that computing technology—one of the most influential defining forces in Western culture of our day, and which shows no signs of slowing down—has no place in education. We are left with a dilemma that I am sure every intellectually honest researcher in the field has had to consider: *we know this stuff is important, but we don't really understand how*. And so *what shall we do*, right now?

It is not that there haven't been (numerous) answers to this question. But we have tended to leave them behind with each surge of forward momentum, each innovative push, each new educational technology "paradigm," as Timothy Koschmann put it.[2]

I hereby suggest that the solution—not to the larger question of what should we do, right now, but at least to the narrower issue of how we can stop being so blinded by the shiny

2. Koschmann's (1996) article, "Paradigm Shifts and Instructional Technology" suggested that there had in fact been a series of incommensurable paradigms (in Kuhn's sense) governing the field; Koschmann was setting up "computer-supported collaborative learning" as the new paradigm.

exterior of educational technology that we lose all critical sensibilities—is to address the questions of *history and historicism*. Information technology, in education as elsewhere, has a 'problematic' relationship with its own history; in short, we actively seek to deny its past, putting the emphasis always on the now and the new and the future. The new is what is important; what happened yesterday is to be forgotten, downplayed, ignored. This active destruction of history and tradition—a symptom of the "culture of no culture" (Traweek 1988, p. 162) that pervades much of technoscience—makes it difficult, if not impossible, to make sense of the role of technology in education, in society, and in politics.[3] We are faced with a tangle of hobbles—instrumentalism, ahistoricism, fear of transformation, Snow's "two cultures," and a consumerist subjectivity.

Seymour Papert, in the midst of the backlash against Logo in schools in the mid 1980s, wrote an impassioned essay that called for a "computer criticism," in the same sense and spirit as "literacy criticism." In that article, Papert wrote of

> ...a tendency to think of "computers" and "Logo" as agents that act directly on thinking and learning; they betray a tendency to reduce what are really the most important components of educational situtations—people and cultures—to a secondary, facilitating role. The context for human development is always a culture, never an isolated technology. (Papert 1987, p. 23)

An examination of the history of educational technology—and educational computing in particular—reveals riches that have been quite forgotten. There is, for instance, far more richness and depth in Papert's philosophy and his *more than two decades* of practical work on Logo than is commonly remembered. And Papert is not the only one. Alan Kay's story, roughly contemporaneous and in many respects paralleling Papert's, is what follows. Since this story is not widely known, let me begin with a brief and admittedly rough sketch of the origins, general direction, and some of the outcomes of Kay's work.

3. We as a society are ignorant of these issues because, in a sense, they can not be made sense of. MacIntyre (1984) makes the much larger case that morality and ethics cannot be made sense of in the modern world, because our post-enlightenment inheritance is but the fragments of a tradition within which these could be rationalized.

Alan Kay and the Dynabook Vision

Alan Curtis Kay is a man whose story is almost entirely dominated by a single vision. The vision is that of *personal computing*, a concept Kay began to devise in the late 1960s while a graduate student at the University of Utah. It is not an overstatement to say that Kay's vision has almost single-handedly defined personal computing as we know it today. Neither is it an overstatement to say that what he had in mind and what we've ended up with are very different. The story of that vision—how it has managed to manifest itself on all our desks (and laps) and also how far this manifestation remains from its original power and scope—is the story I mean to tell here. It is a story that deviates from the popular or conventional story of computing in a number of interesting ways. And, while this story is well known, it is rarely told outside of the computer science community, where Kay's contributions are foundational. What is less remembered is that Kay's contributions to computer science were driven largely by an educational vision for young children.

Alan Kay was born in the early 1940s in New England, and grew up as something of a child prodigy; he proudly reports being a precocious—difficult, even—child in school, arguing with his elementary school teachers. He studied biology and mathematics in university, but dropped out and played jazz guitar in Colorado for a few years in the early 1960s; then, on the strength of an aptitude test, joined the US Air Force and became a junior programmer. Having then discovered computers, he decided to finish his undergraduate degree and go to grad school in 1966. He chose the University of Utah, where computer graphics pioneer Dave C. Evans had set up one of America's first computer science programs. At Utah, Kay's career took off like a rocket; the timely meeting of a wildly creative mind with the fledgling American computing research program—Kay was only the seventh graduate student in computing at Utah (Hiltzik 1999, p. 86ff).

To appreciate the difference between computing as most people encounter it today—personal laptop computers with graphical user interfaces, connected wirelessly to a global Internet, using the computer as an access and production environment to *media*—and what

computing was in the mid 1960s—expensive and delicate mainframe computers staffed by scientists, with little that we would recognize as "user interface" (even time-sharing systems were a radical innovation at that time)—is to roughly frame Kay's contribution to the field. Of course, he did not accomplish this alone, but his vision—dating back to his MSc. and PhD theses at Utah (see Kay 1968) and strongly driving the research of the 1970s—is so central, and so consistent, that it is arguable that without Kay, the face of our everyday involvement with digital technology would be immeasurably different today.

Kay is in one sense an easy study, in that he has remained consistently on point for thirty-five years, over which time he has contributed a large collection of reports, articles, chapters, and postings to online fora, as well as a large number of lectures and presentations, many of which have been recorded and made widely available. In particular, Kay's writings and talks in recent years provide valuable reflection on his work and writings from the 1960s and 1970s; in all, a rich archive for the historian. What I find most important about Kay's *oeuvre* is, I believe, summarizable in a few brief (though rather expansive) points. These set the stage for the story I will attempt to tell here:

- Kay's vision (circa 1968) that in the near future, computers would be the commonplace devices of millions of non-professional users;
- Kay's realization that this kind of mass techological/cultural shift would require a new *literacy*, on the scale of the print revolution of the 16th and 17th centuries;
- his belief that children would be the key actors in this cultural revolution;
- his fundamental approach to the design challenge presented by this shift being one of *humility*, and thus that the cardinal virtues would be simplicity and malleability, such that these "millions of users" could be empowered to shape their own technological tools in accordance with the needs that they encountered;
- Kay's insistence on a set of architectural principles inspired by the cell microbiology and complex systems theory of the post-war period: how the complexity of life arises from the relatively simple and common physics of the cell.

There are many ways in which Alan Kay's vision of personal computing has indeed come to pass. In reading his manifesto from 1972 ("A Personal Computer for Children of All Ages"), there is little that sounds either dated or far-fetched. Most of the implementation details alluded to in his writings have in fact become commonplace—Kay was unable to predict the dynamics of the marketplace on personal computing, and so his timelines and pricepoints are both underestimated. It is indeed clear that his vision of a new "literacy" far exceeds the reality on the ground today. My contention is that this is the piece of his vision which is the most critical; the need for a digitally mediated literacy is greater now than ever, and for reasons which Kay could hardly have foreseen in the early 1970s.

From ARPA to Xerox PARC

Alan Kay's story begins with the ARPA project—the US Department of Defense's *Advanced Research Projects Agency*, a Pentagon funding programme in part inspired by the Cold War and the perceived threat to American technological superiority that was raised with the launch of the Soviet satellite, *Sputnik*, in 1957. In ARPA is the root of the popular conception that computers have sprung from the military; the vast majority of computing research in the formative decade of the 1960s was funded by ARPA's Information Processing Techniques Office (IPTO). It is easy to take the significance of this funding formula too far, however, and conclude that computers were devised as weapons and digital technology is born of insitutionalized violence and domination. The story is quite a bit more subtle than that: the adminstrators of ARPA-IPTO research funds were not military men, but civilians; not generals but professors (NRC 1999; Waldrop 2001). It is perhaps better to think of ARPA as a Cold War instrument of American techno-cultural superiority, rather than a military programme. The funds flowed through the Pentagon, but the research was astonishingly open-ended, with the majority of the funding flowing to universities rather than defense contractors, often in the absence of formal peer-review processes (NRC 1999, pp. 101–102). In fact, to look deeply at ARPA and its projects is to see a ironic case—rare, but certainly not unique (AT&T and, as we shall see, Xerox Corporation, played host to similar

development communities)—of large-scale public works being committed in the name of capitalist, individualistic, American ideology. The *public* funding that went into ARPA projects in the 1960s no doubt vastly outstripped their Soviet counterparts; who, then, had the greater public infrastructure?

The men who directed the ARPA-IPTO have come to be known by their reputation as great thinkers with expansive ideals for the common good, and their open-ended funding policies that focused on people rather than specific goals. The first, and most celebrated director, JCR Licklider, oriented the IPTO to the pursuit of *interactive computing* and *internetworking*, concepts which were nearly science fiction in 1960, but which today are foundational to our dealings with digital media.[4] After Licklider came Ivan Sutherland, known as the "father of computer graphics;" Robert Taylor, who would go on to help run Xerox' research lab in 1970; and Lawrence Roberts, who in the late 1960s oversaw the implementation of the ARPAnet, the prototype and direct ancestor of today's Internet.

In 1970, the advent of the Mansfield Amendment, which required Pentagon-funded research to be more responsible to military ends, is seen by many (Kay 1996*a*, p. 525; Waldrop 2001, p. 325) as the end of an era—an era in which the basic shape of today's digital technological landscape was being laid out. Of the spirit of the ARPA project in the 1960s, Kay reflected:

> It is no exaggeration to say that [ARPA] had "visions rather than goals" and "funded people, not projects." The vision was "interactive computing as a complementary intellectual partner for people pervasively networked worldwide." By not trying to derive specific goals from this at the funding side, [ARPA] was able to fund rather different and sometimes opposing points of view. (Kay 2004*a*)

The legacy left by the 1960s' ARPA project is rich, and includes the Internet, time-sharing systems, computer graphics (both 2D and 3D), hypertext and hypermedia, and networked

4. Licklider wrote an early research manifesto called "Man-Computer Symbiosis" which laid out a blue-sky vision of what computing could become, one in marked contrast to the then-dominant trend to artificial intelligence research. See Licklider 1960; Wardrip-Fruin & Montfort 2004).

collaboration. More important to the story at hand is the establishment of a community of computing researchers in the United States, from universities like Utah, UCLA, Stanford, MIT, and Carnegie-Mellon. At these universities, fledgling computing departments and programs had received early and substantial research funding, and the ARPA-IPTO directors made substantial efforts to bring these researchers together at conferences and retreats. The result was, by the late 1960s, a tightly knit community of American computer science research.

Alan Kay, who had his first encounter with computer programming while on a stint in the US Air Force's Air Training Command in 1961, went to the University of Utah to pursue a Masters degree. There he met and studied with Dave Evans and Ivan Sutherland, who were pioneering research in computer graphics. Kay spent the years 1966–1969 at Utah, working on and around ARPA-funded projects. It was here, in his MSc and PhD work, that he began to formulate a vision for a personal computer. Kay has referred to Sutherland's work on computer graphics as "the first personal computer" because Sutherland's project—*Sketchpad*—was the first interactive graphics program as we would recognize it today; a user sat in front of a display and manipulated the images on a screen by means of a pointing device (in his instance, a pen) and keystrokes (Sutherland 1963). This required that a single user monopolize the entire computer—in the 1960s an enormously extravagant thing to do.

The inspirational impact of work like this should not be understated, especially where Alan Kay is concerned. Kay's account of the ARPA years is of one mind-blowing innovation after another—from Sutherland's elegant drafting program to Doug Engelbart's famous 1968 demo to the Fall Joint Computer Conference in San Francisco which showed the world a working model of hypertext, video conferencing, workgroup collaboration, and graphical user interfaces, literally decades before these concepts became embedded in the public imagination (Engelbart & English 1968/2004; Waldrop 2001, pp. 297–294). Kay's research at Utah focused on the design of a computing system called the *FLEX Machine*, which combined the interactive graphics ideas of Sutherland's Sketchpad with leading programming language concepts of the day and put them in a package that could sit on a desk. But

Kay's work at Utah was very much coloured by interaction and collaboration with the community of ARPA researchers.

> One of the greatest works of art from that fruitful period of ARPA/PARC research in the '60s and '70s was the almost invisible context and community that catalysed so many researchers to be incredibly better dreamers and thinkers. That it was a great work of art is confirmed by the world-changing results that appeared so swiftly, and almost easily. That it was almost invisible, in spite of its tremendous success, is revealed by the disheartening fact today that, as far as I'm aware, no governments and no companies do edge-of-the-art research using these principles. (Kay 2004*a*)

When ARPA's funding priorities shifted to military applications in 1970, this community saw a crisis of sorts; where could they continue their work in the manner to which they had become accustomed?[5] As the story goes, by historical accident, Xerox corporation, as part of a shift in upper management, wanted to establish a research lab to ensure their continued domination (Hiltzik 1999; Waldrop 2001, p. 333ff). As it turned out, former ARPA-IPTO director Robert Taylor was hired on at Xerox to establish the lab and hire its researchers. Taylor knew who he wanted, by virtue of the community of researchers he had known from his ARPA work. And, due to the circumstances of the funding landscape of the day, he had his pick of the leading researchers of the 1960s. The result, Xerox' Palo Alto Research Center (PARC), was staffed by a "dream team" of talent. Former PARC researcher Bruce Horn reflected, "PARC was the Mecca of computer science; we often said (only half-jokingly) that 80 of the 100 best computer scientists in the world were in residence at PARC" (Horn, n.d.).

Alan Kay was one of the researchers that Taylor courted to be part of the Xerox PARC team, and in line with the open-ended ARPA policy of the 1960s, Kay's agenda at Xerox was also open-ended. He took the opportunity to use his new position to advance the work he had begun at Utah on the development of a personal computer.

---

5. Young computer scientists in the 1960s were as attentive as any to the cultural movements of the day; see John Markoff's (2005) *What the Dormouse Said: How the 60s Counterculture Shaped the Personal Computer Industry*, for this treatment.

It was in fact impossible to produce something like Kay's desktop-oriented *FLEX Machine* given the hardware technology of the late 1960s, and as such Kay's early work was realized in various forms on the rather larger computers of the day. But to reduce the *FLEX Machine* concept to simply that of a graphics-capable system that could sit on a desk (or even, ultimately a lap) is to miss much of Kay's point. More fundamental to Kay's vision was a novel and far-reaching conception of computing architecture, and the *FLEX Machine* research is better positioned as an early attempt to articulate this. To explain this, let me delve into Sutherland's *Sketchpad*, a system which in Kay's view has not been equalled in the nearly four decades since. The overt concept—an interactive computing system for drawing and manipulating images—of course has been built upon, and today designers, illustrators, draftspeople, and indeed anyone who creates images with a computer uses a system which borrows from the general tradition established by Sketchpad. But integral to Sutherland's original system was an architecture based on "master" drawings could be used to create "instance" drawings, and that the parent-child relationship between such entities is preserved, so that changes made to the master (or prototype) would be reflected in any instances made from it. It is difficult to express the importance of this in so many words,[6] but this concept is representative of a way of thinking about the relationship between the part and the whole which underlies all of Kay's work and contributions.

At the same time that Kay was introduced to Sutherland's work, he was also introduced to a programming language called *Simula*, the work of a pair of Norwegian researchers. Kay recognized that the "master" and "instance" relationship in Sketchpad was very similar to the way the Simula language was arranged.

> This was the big hit, and I have not been the same since. I think the reasons the hit had such impact was that I had seen the idea enough times in enough different forms that the final recognition was in such general terms to have the quality of an epiphany. My math major had centered on abstract algebras with their few operations applying to many structures. My biology major had

6. There is, luckily, video available of Sutherland using the Sketchpad system. See Wardrip-Fruin & Montfort (2004).

> focused on both cell metabolism and larger scale morphogenesis with its notions of simple mechanisms controlling complex processes and one kind of building block being able to differentiate into all needed building blocks. The 220 file system, the B5000[7], Sketchpad, and finally Simula, all used the same idea for different purposes. Bob Barton, the main designer of the B5000 and a professor at Utah, had said in one of his talks a few days earlier, "The basic principle of recursive design is to make the parts have the same power as the whole." (Kay 1996*a*, p. 516)

This is the first of the "big ideas" that comprise Alan Kay's work; we shall encounter several more.

Toward the Dynabook

Kay reports that his personal trajectory was significantly altered by a visit to see Seymour Papert's research group at MIT in 1968. At that time, Papert, Wally Feurzig, and Cynthia Solomon were conducting the initial research on exposing schoolchildren to computers and programming with the *Logo* language, which Feurzig had designed. Papert's research involved the now-famous "turtle geometry" approach which suggested that children could more effectively bridge the divide between concrete and formal cognitive stages (from Jean Piaget's developmental schema) via a computational medium (Logo) which allowed them to manipulate mathematical and geometric constructs concretely (Papert 1980*a*; 1980*b*). What impressed Kay was not so much this insight about cognitive styles, but that children using Logo could reach farther with mathematics than they could otherwise. Kay wrote:

> One of the ways Papert used Piaget's ideas was to realize that young children are not well equipped to do "standard" symbolic mathematics until the age of 11 or 12, but that even very young children can do other kinds of math, even advanced math such as topology and differential geometry, when it is presented in a form that is well matched to their current thinking processes. The Logo turtle with its local coordinate system (like the child, it is always at

7. The Burroughs B220 and B5000 were early computers Kay had encountered in while working as a programmer in the US Air Force in the early 1960s.

> the center of its universe) became a highly successful "microworld" for exploring ideas in differential geometry. (Kay 1990, p. 194)

In what would be the beginning of a collegial relationship with Papert which is still ongoing, Papert's insights about children and computers, in combination with Kay's insight that computers would likely be much more numerous and commonplace by the 1980s, led to the crystallization of his thinking:

> This encounter finally hit me with what the destiny of personal computing *really* was going to be. Not a personal dynamic *vehicle*, as in Englebart's metaphor opposed to the IBM "railroads," but something much more profound: a personal dynamic *medium*. With a vehicle one could wait until high school and give "drivers ed," but if it was a medium, it had to extend to the world of childhood. (1996*a*, p. 523)

Kay was immediately seized by this idea, and on the plane back from Boston he drew up the basis for the vision of personal computing he would pursue thereafter. Kay called it the *Dynabook*, and the name suggests what it would be: a dynamic book. That is, a medium like a book, but one which was interactive and controlled by the reader. It would provide cognitive scaffolding in the same way books and print media have done in recent centuries, but as Papert's work with children and Logo had begun to show, it would take advantage of the new medium of computation and provide the means for new kinds of exploration and expression.

Kay, now at Xerox PARC, began to sketch out what the Dynabook would look and act like. Early models (in cardboard) suggest devices not unlike the desktop and laptop computers we know today. Kay noted that he was directly inspired by (microchip manufacturer Intel's founder Gordon) Moore's Law, which states that, due to predictable advances in the miniaturization of chip manufacturing, available computing power *doubles* roughly every 18 months. Given this kind of development timeframe, Kay foresaw that by the 1980s, the sorts of things he was able to accomplish in his work on the *FLEX Machine* would indeed be possible on small, even portable devices.

Again, however, it is important to sidestep the temptation to reduce Kay's vision to a particular conception of a hardware device or set of features. The deep levels of his research were aimed at coming up with ways in which people—not computer scientists, but schoolchildren, after Papert's examples—could interact meaningfully with digital technology. In the 1960s, computers were still monolithic, vastly expensive machines; leading research of the day was aimed at the development of "time-sharing" systems which would allow multiple users to simultaneously use a large computer by connecting via a *terminal*—this was profoundly not "personal" computing. Despite the economic and logistical obstacles, Kay and his newly established *Learning Research Group* at Xerox PARC wrestled to come up with a new model of how people could interact with computing technology. Kay's reflections on the challenge give some sense of the scope of the task they set themselves:

> For example, one would compute with a handheld "Dynabook" in a way that would not be possible on a shared main-frame; millions of potential users meant that the user interface would have to become a learning environment along the lines of Montessori and Bruner; and needs for large scope, reduction in complexity, and end-user literacy would require that data and control structures be done away with in favor of a more biological scheme of protected universal cells interacting only through messages that could mimic any desired behaviour. (Kay 1996*a*, p. 511)

This was research without precedent in the early 1970s—no existing models of computing or user interface or media were extant that Kay and his group could follow. In a sense, it is interesting to think of the work done in the 1960s and 1970s as being *groundbreaking* because of the the lack of existing models. Kay recalls the number of innovative conceptual leaps that Ivan Sutherland's *Sketchpad* project made, and that when asked how this was possible in 1963, Sutherland later reflected that he "didn't know it was hard."

What Kay and his colleagues seem to have been aware of is the sense in which they were in fact constructing whole new ways of working:

> ...we were actually trying for a qualitative shift in belief structures—a new Kuhnian paradigm in the same spirit as the invention of the printing press—

> and thus took highly extreme positions that almost forced these new styles to be invented. (1996*a*, p 511)

The analogy of the printing press is one that bears more examination, if for no other reason than that Kay himself has extended the analogy. If the invention of the digital computer can be compared with the invention of the printing press, then it follows that there is an analagous period following its initial invention in which its role, function, and nature have not yet been worked out. In the history of printing, this period was the late 15th century, commonly called the *incunabula*, when early printers experimented with ways of conducting their craft and their trade. The earliest printers, like Johannes Gutenberg, created printed works that closely mimicked the work of medieval scribes in design, content, and audience. As Marshall McLuhan (1965) noted, the content of the new media is the old media.

But in the 1490s, the Venetian printer Aldus Manutius set up a printing business which, in its exploration of the possibilities of finding a sustainable business model, pioneered and established much of the form of the book as we know it today, in terms of layout, typography, size and form (Aldus is generally credited with the popularization of the *octavo* format, which would fit conveniently in a pocket or saddlebag), and, in doing so, defined a new audience and market for printed books (Lowry 1979). Aldus' innovations established the nature of the printed book as we know it today, and innovations in book printing since then have been refinements of Aldus' model, rather than deviations from it.

Alan Kay alludes to the example of Aldus in several places in his writings, and it seems clear that even if Kay doesn't necessarily consider his work to be parallel, then at least this is its goal. That we are in an *incunabula* period in the early evolution of digital computing—as evidenced by the general confusion the topic evinces—is an idea I am completely comfortable with; that Alan Kay's vision of personal computing is analogous to Aldus' pocket-sized books is at least worth consideration. Whether we can say one way or another, at this moment in time, is in part the subject of this dissertation.

Elements of the Dynabook Vision

In a paper presented in 1972, "A Personal Computer for Children of All Ages," Alan Kay spoke of the general qualities of a personal computer:

> What then is a personal computer? One would hope that it would be both a medium for containing and expressing arbitrary symbolic notations, and also a collection of useful tools for manipulating these structures, with ways to add new tools to the repertoire. (Kay 1972, p.3)

Papert's influence is very clear here, especially his famous admonition that *children should program the computer rather than the computer programming the children*. But we should also pay attention here to Kay's emphasis on the multiple levels of media: that they should represent not just the content, but the tools to act upon the content, and even the means for creating new tools. This sheds some light on the Dynabook metaphor, for books represent not only content which can be extracted (as a shallow definition of literacy might suggest), but are also the means to participating richly in a literate culture. "One of the primary effects of learning to read is enabling students to read to learn" (Miller 2004, p. 32). Literacy is indeed what Kay and his team were after. "I felt that because the content of personal computing was interactive tools, the content of this new authoring literacy should be the creation of interactive tools *by the children*" (Kay 1996*a*, p. 544, italics mine).

Kay's 1972 paper included a scenario in which two nine-year olds, Jimmy and Beth, are playing a video game,[8] "lying on the grass of a park near their home." Young Beth, bored of repeatedly trouncing her classmate, muses about adding gravitational forces to the game in order to make it more challenging. The rest of the story has the two children seeking out their teacher to help them develop their model of how the gravitational pull of the sun should be integrated with the spaceship controls in the game. Together, "earnestly trying to discover the notion of a coordinate system," they use something much like the Internet to look up some specifics, and then Beth makes the changes to the physical model coded in the game. Beth later uses her Dynabook to work on a poem she is composing, and her father, on

8. The game is the prototypical *Spacewar!*, which has a special place in the history of computing.

an airplane on a business trip, uses his own Dynabook to make voice annotations to a file, and even to download (and neglect to pay for) an e-book he sees advertised in the airport.

Kay was writing science fiction; it was 1972. But the vision is clear enough that we can easily recognize almost all of these elements in our quotidian computing environment.

> It is now within the reach of current technology to give all the Beths and their dads a "Dynabook" to use anytime, anywhere as they may wish. Although it can be used to communicate with others through the "knowledge utilities" of the future such as a school "library" (or business information system), we think that a large fraction of its use will involve reflexive communication of the owner with himself through this personal medium, much as paper and notebooks are currently used. (1972, p. 3)

Most importantly, Kay was close enough to the cutting edge of computer research to be able to judge just how "within reach" this vision really was. Kay's oft-quoted catchphrase, "*the best way to predict the future is to invent it*," meant that his science-fiction writing was nigh on to a plan for implementation. His work at Xerox through the 1970s was nothing short of the realization of as much of the Dynabook plan as possible at the time. Kay foresaw that, given Moore's Law, that the hardware part of his vision should be feasible within a decade or so: the 1980s. The unknown part was the software. So, while much of the famous research at Xerox PARC was in producing the first "personal computers" (these were hardly laptops; they were however small enough to squeeze under a desk); Kay's core focus was on the software vision; how would Beth and Jimmy actually interact with their Dynabooks? How would millions of users make effective use of digital information technology?

What emerged after a few design iterations in 1971 and 1972 was a programming language called *Smalltalk*,

> as in "programming should be a matter of..." and "children should program in..." The name was also a reaction against the "IndoEuropean god theory" where systems were named Zeus, Odin, and Thor, and hardly did anything. I figured that "Smalltalk" was so innocuous a label that if it ever did anything nice people would be pleasantly surprised. (1996*a*, p. 528)

Smalltalk was (and is) a programming language; the original version—implemented the following year and therefore designated *Smalltalk-72*—owed much to Papert's Logo in terms of syntax and aesthetics. But its aspirations were considerably greater—in many ways, it was a generalization of the sort of thing Papert was after. Kay went so far as to eschew the "programming language" description, instead calling Smalltalk "a new medium for communication" (Kay & Goldberg 1976).

Kay's research—and Smalltalk itself—got a boost in 1973 when researchers in PARC's Computer Science Lab developed the first iterations of the *Alto* workstation, which is commonly hailed as the first personal computer.[9] Kay and his team called the Alto an "interim dynabook"—not much like Kay's Dynabook vision at all, really—these were about the size of a bar-fridge—but the Alto is the direct precursor of the kinds of personal computers we have today (as opposed, that is, to the personal computers of the late 1970s and early 1980s): it had a bitmapped, graphical display, a pointing device, and, with Smalltalk running on it, the kind of "desktop" environment we now take for granted. In 1973–74, these "interim dynabooks" were capable of Logo-like turtle graphics, but also featured a mouse-and-overlapping-windows interface, animated graphics, and music—in short, "multimedia."

At the same time that the Alto was being created, and that Kay's team was working on Smalltalk, other researchers at Xerox PARC were developing *ethernet* local-area networking, colour graphics software, word processing and desktop publishing, and the laser printer—all commonplace components of "personal computing" today, and all fitting neatly into Kay's Dynabook vision—what Kay would much later call the "PARC genre" of computing (2004*a*). The researchers at Xerox PARC created hundreds of Altos and wired them all up with ethernet, installed precursors to office software, and had, by the mid 1970s, an internal prototype of the kind of computer-based office environment that is so commonplace today. None of this would be commercialized or marketed until the following decade, but it was all running at PARC, and there it was allowed to mature into an established

9. Reportedly, over 1500 of these machines were constructed and used by individuals at Xerox in the 1970s (Hiltzik 1999).

pattern of information technology. And, of course, many of the researchers at PARC in the 1970s subsequently left to start or join the companies that now dominate the computing world (3Com, Adobe, Microsoft, Apple, etc.).

To end the story here is to uncritically accept Alan Kay's popular designation as "father of the personal computer." But what is missing from contemporary tellings of this heroic tale is the seed that started it: the vision of children and a new literacy. Notably, the three best available histories of this period (Smith & Alexander 1988; Hiltzik 1999; Waldrop 2001) significantly downplay or pass over the educational vision which provided the focus for Kay's work. It is easy enough to see ourselves as adult, even professional users of desktop computing systems like those pioneered at Xerox PARC, but where are Beth and Jimmy today, "earnestly trying to discover the concept of a coordinate system?"

The fate of the Dynabook

The personal computer did indeed come to be, very much as Kay anticipated. Indeed, I have written the present work on a notebook computer strikingly similar to the one Kay described in 1972. His vision of millions of users of computers is very much the reality today. But, I argue, the Dynabook vision *has not been realized*; the distinction I am making here is between the idea of portable, networked personal computing devices on the one hand, and the vision of a new literacy and attendant educational imperative on the other. Between the surface features of a personal computer and Kay's deeper insights about what that personal computing should entail is a vast gulf. The difference is significantly not one of technological innovation; all the individual components of Kay's vision are extant, even mature technologies today, from the lightweight, wirelessly connected notebook computers equipped with multimedia authoring tools to the kind of simple, modular software models he pioneered (indeed, this part of the vision has been picked up by computer programmers and turned into a whole paradigm of software development). Rather, the difference is a cultural one, wherein what personal and educational computing *means* to us is vastly different from the vision Kay and his colleagues began to elaborate in the early 1970s. We have

inherited all the components, but the cultural framework which ties them together relies on older ideas, and the new computer-mediated literacy that Kay articulated continues to elude us.

The ramifications of this cultural difference are, I argue, vast, and they specifically underlie the problematic relation we have with digital technology I outlined in the early pages of this chapter. The case I will make in the pages which follow is that our contemporary condition of fear and loathing of digital technology, our narrow and ahistorical perspective on computing, our unquestioned acceptance and reification of the roles of 'expert' and 'end-user', and, most importantly, the compounded manifestation of all of these features in the confused world of educational technology can all be critically addressed—and in some cases remedied—by attention to this particular yet foundational thread in the history of computing. Most of us are, unfortunately, starting from a place of unfamiliarity with this tradition (or, for that matter, any such substantial historical perspective); it is my intent with the present study to at least shed some light on a specific cultural tradition which, I believe, has much to say to our current predicaments.

What Follows...

The chapters and pages to follow comprise a historical treatment of Alan Kay's vision and research, of the Dynabook vision and its various (and partial) implementations in changing contexts over three or more decades. I intend here to draw attention to the features of the original vision which have changed through diverse times and contexts, those features which have remained constant, and those which have grown even more critical. Despite Alan Kay's centrality to this story, and his predominant role in its articulation, this is not a biographical treatment; rather, this is an attempt to trace the threads of a technocultural system over time and place. It is thus a work of cultural history.

In the chapter that follows this one, I frame my positionality and approach to the interpretation and historiography of this subject. I begin with my own story, and how by degrees I have come to appreciate the importance and centrality of Alan Kay's contributions, and I

elaborate the many forces which have led to my adoption of a particular attitude to and stance toward digital technology and its place in education. This treatment of my 'methodology' is more a disclosure of my personal biases and theoretical inclinations than an analysis of social scientific method, for reasons I will elaborate in due course.

In the third chapter, I provide a broad-strokes theoretical approach to the study of technology which serves to ground the kind of analytical moves which come later in this account. Here, I address the main motifs of technological *mediation*, the sociology of *translation* (after Callon and Latour), a *semiotics* of digital 'machines,' and finally an introduction to the notion of *simulation*, which I take as the paradigmatic modality of digital media. These treatments set the stage for the history that follows.

My account of the history of Kay's project and the Dynabook vision begins in Chapter 4, in which I conduct a high-level review of the conceptual content of Alan Kay's numerous writings and talks. This review breaks down into six primary themes which I believe reasonably represent the core of Kay's vision of personal and educational computing.

In Chapter 5, I cover in narrative form the key moments of the Dynabook's development through the 1970s and 1980s. This begins with an exploration of the ways in which the Smalltalk language and environment was translated from an educational platform to a profound and influential innovation in professional computer programming and software engineering. Second, I discuss the emergence of a personal computer industry and market in the late 1970s and 1980s and the intersection of this trend with the foundational research work done at Xerox PARC. Third, I trace Alan Kay's research beyond Xerox PARC to its home at Apple Computer in the 1980s, where very different economic, technical, and cultural forces were at play.

Chapter 6 considers what personal computing—and, by extension, educational computing—came to mean in the 1990s, with the popular advent of the Internet and World-Wide Web. In many ways, this period represents the mainstreaming of a particular version of personal computing and a much more substantial cultural tradition against which the Dynabook vision must now be considered. The history of computing in the 1990s

is one of global-scale market developments (i.e., the computing industry as a multi-billion dollar phenemenon), as well as the emergence of unprecedented forms of cultural expression and digitally mediated social organization (and especially, the rise of the Free and Open Source Software movement, arising out of marginal computing cultures from the 1970s); these two large-scale trends are in considerable tension with one another.

In Chapter 7, I trace the actual re-emergence of a substantial chunk of Alan Kay's work and indeed the Dynabook vision against the backdrop of late 1990s computing culture and the trends introduced in Chapter 6. The *Squeak* environment, emerging from Kay's team at Apple Computer in 1996, lays technical and mythological claim to the Dynabook tradition of the 1970s. Here, I examine the development of Squeak, its development and user communities, and I attempt to evaluate its contemporary trajectories. Because Squeak provides an artifact so unambiguously connected to the idealistic work emerging from Xerox PARC in the 1970s, it is possible to interrogate the relative health and coherence of the cultural traditions which it simultaneously draws upon and, arguably, creates anew.

In the final chapter, I attempt to draw things together, bringing the focus back to the macro level and examining the Dynabook ideal in the large: is it better considered as a technocultural artifact or as a touchstone idea connecting otherwise disparate embodiments? This question leads to a broader methodological question, regarding how 'ideas' are to be treated alongside more concrete material objects like artifacts and texts. Finally, I return to the higher level political and social questions of the ultimate relevance of this story to education and to the popular culture of technology.

Chapter 2: Positions and Approaches

Introducing Myselves

In a number of respects I am a child of Alan Kay's vision. I've grown up in a world in which Kay's vision has always to some extent existed, though it was not until recently that I had any sense of this. My own lifetime is almost synchronous with Kay's project; I was born at just about the time that Kay was working on his initial design for a "personal" computer, the FLEX machine, while at the University of Utah. When Kay went to Xerox in 1970 and began to work on a model of personal computing for children, I was just coming to the age at which my education, and my relationship to media, was beginning. Speaking strictly temporally, my generation was the one that Kay was looking at as the target for his notion of personal computing, though it has taken me thirty years to recognize it.

That I have grown up in a world in part defined by Kay's work, and that I have been at least somewhat aware of this fact is key to the present study. The significance of Kay's project, in its educational and political aspects, is something apparent to me perhaps because of the particularities of my own history. In attempting to present my treatment and interpretation of this project and its importance to the world, I am in a position of needing to examine and establish just what it is about my own perspective that makes these issues meaningful for me.

The perspective(s) I present here is not that of a schooled computer scientist or technologist; neither is it as a 'humanist' approaching the history of computing from without. Rather, I claim partial roots on both sides of this seeming divide, and as a result the story I will tell is not likely to be generic in either mode.[1]

1. The evocation of distinct genres of technology historiography and critique is deliberate. Among writers who have attempted to interpret digital technologies to a wide audience, the 'distanced humanist' stance is well exemplified in Edwards (1996); Cuban (2001); Bowers (2000); Menzies (1996); Rose (2003), while the schooled technologist perspective is found in Papert (1980a); Winograd & Flores (1986); Harvey(1991); Stallman (1998); diSessa (2000).

Roots

As a child, I had some early exposure to computers and computing. My father had long been an electronics hobbyist, one of a generation of early radar technicians in the Second World War[2] and a "ham" radio operator since the late '40s or early '50s. I grew up watching him wield a soldering iron, delighting in home-brewing his radio gear. In the 1970s, when microprocessors and integrated circuits ("chips") became widely available, Dad began experimenting with building computers. I remember him working for years on a teletype project—it must have gone through several versions—to be able to type text on a little TV screen; certainly not impressive to our 21st-century eyes, but he was building it almost from scratch, out of parts ordered from the back pages of hobbyist magazines, and he reveled in the pure challenge of figuring out how to make the thing work. Had he really wanted a teletype for his radio gear, he could have bought or wired up a kit in short order, but instead, he worked for some number of years on designing and creating his own. I remember him bringing assembly code written out in pencil on envelopes and notepads to the table with his morning coffee, and also his experimentation with several different modes of creating circuitry: wires on pegboards, and later "etching" his own printed circuit boards.

Despite any or all of this, I was not an electronics "whiz" as a child; I was not at all interested in radios or computers, and while I casually shared in my Dad's intellectual journeys on occasion, I myself was barely competent with a soldering iron, and never learned the workings of the computers he created or how to program them. When I was 12, Dad encouraged me to study up on basic electronics and the Morse code and to take the tests to get my own ham radio license. I accepted the challenge in good spirit, but when the time came I failed the tests—it simply was not compelling to me. I wasn't particularly disappointed, and I remember Dad praising me for being a good sport about it; he never pushed me in that direction again. Comfortable with this arrangement, I continued to watch over his shoulder and be his sounding board as he talked out his design ideas and railed against

2. Dad joined up with the Canadian services in 1941 and was shipped to the UK, where he worked with the British RAF radar corps until the end of the war.

the numerous conceptual obstacles he encountered. I recall him explaining the basics of assembly code programming—learning to work with hexadecimal numbers, add this byte to the register, jump to this location, and so on—but it retained for me the character of an introductory lecture: heard and even appreciated at the time, but quickly filed away and largely forgotten.

Still, the "spirit of the quest" made an impression on me, and the exposure to the conceptual underpinnings—if not the details—of computing has surely stayed with me. In junior high school in the early 1980s, when the new Apple II computers came out, my friends and I played at programming simple things in BASIC, and I talked dad into getting a "real" computer at home (a tiny Sinclair ZX81). I got reasonably good at programming in BASIC, and wrote endless variations on an obstacle course game, littering my programs with clever details, elaborate introductory sequences, and the like—this kind of adornment was what was missing for me from Dad's earlier projects, I suppose. I even became something of a "whiz" among my friends at school, owing to the extra time I had to explore at home. When, in grade 11, I was finally able to take a "computer science" course, I was well ahead of my classmates. Or at least, *most* of my classmates, for there were a couple of boys, whom I didn't know (I honestly wondered where they'd come from) who were clearly years ahead of me in understanding and proficiency. While I could write programs in BASIC, these kids were far beyond that, programming the assembly code my Dad had worked in, devouring acres of information from books and magazines. And I was intrigued, and got to know these kids a little, and learned from them. But shortly something interesting happened: whether it had to do with high school and my nascent awareness of social status, or whether it was a subconsious reaction to not being the head of the class anymore, I decided quite clearly and purposefully that I didn't want to be part of that group; that the obsession that these couple of boys displayed did not strike me as *healthy* in some sense. Around the end of my Grade 11 year, I quite sharply turned away from computers and everything to do with them. The next winter I started playing the bass guitar and I effec-

tively forgot that computers existed. Dad continued with his projects, and my relationship with him had more to do with fishing trips and playing music.

I had nothing to do with computers until my 4th year at university, studying cultural anthropology. I was writing an undergraduate thesis, and I had by now seen enough of my friends writing with a word processor to know that this was clearly the way to go about it. I talked my parents into subsidizing the purchase of an Atari ST machine—billed as a poor-man's Macintosh, as it had a mouse and a graphical interface, and cost about half what a Mac did. The Atari did its job with my undergrad thesis (I am still loathe to throw out the floppy disks containing it), but it did other things as well: the one that made the biggest impression on me was the inclusion—in the bundle of software and toys that came with the Atari—of a version of Logo, the language that Seymour Papert had designed for kids in the late 1960s. I didn't know what Logo was, particularly, but there was a one-page printed reference guide to the language primitives, so I was able to poke away at it and draw pictures on the screen with 'turtle' commands.

I remember very clearly my series of discoveries with Logo; I quickly learned that you could draw stars with it, if you had the turtle move forward a standard distance, then turn some divisor of 720, and repeat this the right number of times. I quickly learned that you could generalize it: draw a line, then turn 720/*n* degrees, and do it *n* times. Gee, I discovered, 360/5 would draw a pentagon, while 720/5 would draw a star—how about that!

I have to admit I was amazed (and I still am); here I had learned something 'real' about geometry that 12 years of school hadn't really made me understand. Obviously, after 12 years of math class I could have told you that the angles of a polygon have some relationship to 360, but the fact didn't really *mean* anything to me until I started playing with Logo. Much, much later I discovered that this was *exactly* the sort of experience that Papert was shooting for—only he had hoped that elementary-school kids would be making this discovery, not undergraduates in their final year. Well, better late than never. I have to give credit to Papert and his team—and this foreshadows an important theme of Alan Kay's which I will return to at length—for they had managed to *embed the potential for a very particular*

kind of experience in their software (twenty-five years earlier, for heaven's sake) so that I could pick it up, sight unseen, almost completely by accident, and immediately have that very experience.

I finished the thesis, graduated, and went off to be a young pseudso-intellectual, spending a lot of idle time talking with my friends about ideas. One of the topics we talked about at length was computers and the idea that people would soon—perhaps already were—inhabit a computer-mediated environment. We all read William Gibson's prophetic novels about cyberspace and were enthralled by this notion, almost entirely fictional at the time. In 1992, after a few years of not particularly making a living as a musician, I started looking around for a graduate program that would allow me to explore some of the ideas we had been talking about: cyberspace, hypertext, and so on. I didn't find a graduate program *per se*, but I stumbled upon a one-year diploma program in "Applied Information Technology" at a local community college. I quit my band, bought a Macintosh, and enrolled. What I had little sense of just then was how many of my peers were doing the same thing—quitting their indie rock bands and getting into new media. A few years later I recognized this as a major shift for my generation.

My Encounter(s) with Objects

My return to school in 1992 was the beginning of a series of encounters with Alan Kay's intellectual legacy, encounters which have fundamentally shaped my way of looking at technology and the world I live in. My decision to enroll in the program at Capilano College was mostly serendipitous; I had no defined goals, but I saw a direction to go in. My classmates and I learned about—and produced—educational multimedia game/environments. The dominant software paradigm for us was Apple Computer's *HyperCard*, which, despite its superficial limitations—unmodified, it was only capable of black-and-white presentation—it seemed almost infinitely flexible, at least to my novice understanding. HyperCard was indeed an extraordinarily powerful media production tool, far more so than the bulk of what we have seen on the Web in the past decade. While building and programming multi-

media productions in HyperCard, my friends and I enjoyed 'discovering' an aesthetic of modularity in design—or, perhaps more accurately, HyperCard 'taught' this to us, by way of its elegant conceptual model.

At Capilano College I also first encountered the Internet and the then-tiny World-Wide Web. The Capilano College program required all students to participate in an early computer conferencing system, and this had a minimal Internet connection. We were positioned perfectly to be able to watch the Internet's transformation from something 'behind the curtain' to a massive social force over the space of a few years.

In 1993, when I graduated from the Infotech program, I printed up a set of business cards that said, "John Maxwell – Hypermedia Architect." Paying work didn't really arrive for some time, and I spent a very lean few years in the mid 1990s, buoyed up by my raw enthusiasm, with lots of time on my hands to immerse myself in the developing Internet culture, learn new things, and talk at length with my small circle of friends. We were hopelessly idealistic, and formed a co-operative to share work and forward the ideals of a non-commercial, arts-oriented digital world, rejoicing in an early version of online culture in which business motives were scorned in favour of a kind of techno-romanticism.

In those years, while the Internet and the World-Wide Web were still largely fluid, undefined spaces (rather like my career), I spent a good deal of time mucking about with somewhat marginal technocultural oddities called MUDs—*multi-user dungeons*. A MUD is a real-time, networked, multiple-participant, text-based virtual reality (Curtis 1992). The beginnings of the MUD phenomenon were in the early Dungeons and Dragons-inspired "adventure" computer games. A MUD is, at its simplest, a text-adventure game which can accomodate more than one player; two or more people can thus gang up on the dragon, or what have you.

My interest in MUDs wasn't the game aspect—I have never been a computer-game player—rather, I had been introduced to a particular Internet MUD called *LambdaMOO* by a friend who was very immersed in Internet culture. LambdaMOO was an immense, free-flowing social environment, like a sprawling text-based house party[3]. The critical innova-

tion of LambdaMOO was that the virtual environment was entirely constructed *from within*, by its players, rather than by a specially empowered designer/programmer. LambdaMOO when I first encountered it was only a year or two old, but its 'topography' was already immense and complex, simply because some thousands of users from all over the Internet had been constructing and programming it. LambdaMOO was, I learned, the pet project of a computer scientist at Xerox PARC (which I had never heard of); its language and internal architecture were "object-oriented"—a term I had heard but which had little meaning for me—hence "MOO," for *MUD, Object-Oriented*.

In practical terms, what this meant was that you could create new things and define their behaviour in the virtual world by *basing them on already existing virtual objects* and then specializing them by writing simple scripts. This meant that individual players could very easily create complex, interactive objects within this virtual world. One could easily recognize a kind of aesthetic of creation in MOO worlds. The artfulness of it was a particular kind of *illusionism*: for instance, considering the best way to create a green grassy space in the virtual environment brought into sharp relief heady issues of Platonism, simulacra, and phenomenology. Though I have never done any theatre, its connection and relevance to MOOing was obvious.[4] I thought MOO was fabulous! At the time, I honestly felt that this was—in spite of its completely text-based interface—a vastly more important and promising technology than the World-Wide Web: here were people presenting themselves and their environment virtually, in time. How much more interesting than static web pages!

Fed up with my impoverished experience as a reluctant Internet entrepreneur, I went back to school again to take a new graduate program in publishing. Part of the program was a 4-month applied internship, and I set up a project working with a distance education program at BC's Open Learning Agency (OLA). The project was to create a MOO-based

3. LambdaMOO is still running—at lambda.moo.mud.org:8888—more than fifteen years after it opened, which must make it one of the longest-running persistent object stores in existence. The original virtual space has been added on to tens of thousands of times by tens of thousands of 'players,' but the living room into which you emerge from the darkened 'coat closet' remains the same, as does the inanity of the conversation one can find there, any time of the day or night, any time in the last decade and a half.
4. I later learned that Juli Burk (1998) explored this theme in some detail.

environment for high school distance learners. Here was an opportunity to do what I really wanted: to immerse myself in a project, technically and culturally, and to do some high-level reflection. The result was my Masters project (Maxwell 1996), one of the most rewarding things I have ever done.

The Open Learning Agency was a good fit for me, and I stayed on after my internship. Ironically, I was soon involved in a project which would have been a much better fit for a publishing internship than my MOO project had been: the OLA's schools program was interested in offering their high-school distance education courses *online* as well as *in print*. The challenge of how to do both without having to do it *twice* was paramount, but a friend of mine there, Prescott Klassen, had an answer that set the direction for the next three years of my life. The answer to the problem of publishing in two formats from a single editorial process was a document management technology dating from the 1970s and 1980s called Standard Generalized Markup Language (SGML).[5] Klassen and I embarked on an ambitious project to design and implement an SGML system and workflow to completely overhaul the OLA's courseware production.

The SGML project was, in retrospect, a descent into the abyss, but many good things came out of it. The project was technically a success, but organizationally doomed, and I gained a wealth of insight into the cultural dynamics of technology integration. I also learned a lot in those three years—easily more than in any other period of my life—about computing, document management, and publishing technology. Not surprisingly, Klassen and I, having been given more or less free rein to do what we wanted, were able to move much faster than the organization (a working group of 25 or so) we were attempting to change (Klassen, Maxwell, & Norman 1999). At the end of three years, our proof-of-concept complete, we both left the OLA burnt out (along with a handful of other people close to the project, including our director). Prescott went to work on similar publishing projects at Microsoft Press, and I went back to school.

5. SGML is an ISO standard for document management (see Goldfarb & Rubinsky 1991). A somewhat bastardized application of SGML is found in the Web's HTML technology.

The lasting pieces of the SGML project at OLA that I took with me are these: my gradual appreciation of the *history* of computing; that in many, many cases, the solutions to today's challenges are to be found in technology developed in the 1960s or 1970s, and that these solutions are very often the work of individuals or small teams of thinkers—that as a result, they are *wholly graspable* by an individual, given a commitment to explore not just the details, but the historical contexts of their development. The larger theme lurking in this issue is that there exists a tradition of "powerful ideas" in computing, ideas which are unfortunately often ignored or bypassed in favour of the superficial, partially understood implementations, which constitute another, larger tradition of discourse and practice. To see through the partial, ahistorical implementations to the clearer thinking and powerful ideas lurking behind them gives one, first, a more solid place from which to withstand the constant churn and instability of the market-driven world of IT, and second, an appreciation that many of the big ideas in the history of computing are about people and cultures first, and the details of technical application second. I began to believe that most of this "isn't rocket science" after all. The possibility of demystification was enormously empowering.

A particular case of this comes to mind; at a document-technologies conference in 1998, I witnessed two of the leading minds in the field of SGML and XML—Tim Bray and Eliot Kimber—engage in a debate about the role of abstraction in information representation. Kimber, a proponent of an abstract tree-based information architecture called "groves," argued for independence from and priority to particular representational strategies (like SGML or XML—see Kimber 1998). His admonition: *"transcend syntax."* Bray, on the other hand countered by appealling to the Unix tradition and way of doing things, claiming that by agreement on a simple representational format (e.g., simple structured text files), a great arsenal of software tools could be combined in ways not foreseeable by the original architect. This debate, I realized, was one of cultural difference rather than of technical merit. That realization led me to understand that at the core of technological systems lay people and practices, and that it was an understanding of *these* that was important. This

does not—emphatically does not—mean that the technical components of a system are irrelevant, or interchangable, or governed by social factors; following Latour, it means that the technical facets can only be understood *well* by seeing their embeddedness and participation in historical/cultural traditions of thought and practice. I was, however, sufficiently well steeped to appreciate both Kimber's and Bray's arguments in this light, rather than getting lost in the 'technical' details.

The aforementioned "groves" concept on its own made a huge impression on me, as well, though I am convinced now that I only scratched the surface of it. In simple practical terms, it means the representation of a document or any document-like information as a tree structure, which can then be topologically treated in software. That such a structure can be simultaneously and reciprocally *abstracted* and *concretized* was another encounter with object orientation; that by abstraction, we gain an alternate realization. This is difficult to describe in so many words, but my *seeing* this concept[6] meant that I would never see a 'document' the same way again. For instance, it immediately led to a further realization that my object-oriented grassy field in LambdaMOO and the semantic structure of a term paper were made of the same kinds of things, and were, in some fascinating ways interoperable. Note that I am not talking here about the level of bits—of the ones and zeros that make up computation at the lowest level—rather, I mean this in terms of high-level structure, at the highest semantic levels rather than the lowest.

This was my intellectual journey, at least. At the same time I was discouraged and fed up with 'distance learning' and its institutional evolution[7]—which was core to our work at OLA and which was just beginning to become a major area of interest in the late 1990s. In 1999 I went back to school, (for good this time) and began a PhD program in education. But I wanted to avoid the "educational technology" I had been working at OLA. Instead, I preferred to spend time reading and thinking curriculum theory, continental philosophy, and

6. One must bear in mind while reading my pained attempts to render my own experience of this into words that I am not a mathematician, that I have always had a very strained and compromised relationship with mathematics. I dearly wish that my mathematical education had been better, for I have travelled this far in my life almost despite it.
7. David Noble's 1999 "Digital Diploma Mills" is singularly instructive on this point.

decentered ethnography (while studying with Ricki Goldman). My techie side, all the while, was paying the bills by means of a few sizable technical contracts for the OLA, creating a couple of different iterations of a learning environment which would provide personalized access to "learning objects" (since our SGML project had left the OLA with a large store of curriculum content easily broken up into such modular components). I had, at a deeper level, ceased caring seriously about "learning objects,"[8] but since OLA (and every other educational institution in the networked world) was interested, these were the raw material for our efforts. Of more practical and intellectual importance to me was adopting an object-oriented platform for our development work.[9] More importantly, however, this work brought me incrementally closer to figuring out what object-oriented programming and systems were all about, and *where this tradition had come from*. By degrees, I began to realize that there was a common intellectual thread underlying almost every significant idea I had become aquainted with in the past decade: from HyperCard's elegant authoring environment to MOO's build-from-within model, document "groves," the contextualization and re-contextualization of learning objects, and the abstractions we were able to make designing online information spaces.

The catalyst to my conceptualizing all of this as a common thread was my discovery of a paper written by Alan Kay for the *History of Programming Languages II* conference in 1993 (Kay 1996*a*) on "The Early History of Smalltalk." This paper, positioned amid a collection of highly technical reports on computer science history, stood out at once for me. In the first place, Kay was talking about an educational project for children, rather than the design of systems for professional programmers. Second, the sheer conceptual scope of the article—ranging from historical treatments of printing and print literacy to the evolution of computer graphics and interface design; from the ARPA community in the 1960s to the design of

8. For the most part, "learning objects" are related to object-oriented design in name only. The very idea of "learning objects" as portable, recombinable curriculum components is, I think, highly suspect. Friesen (2001) is as good a 'last word' on this topic as I have seen.

9. That platform was Zope, an open-source object-oriented web publishing application, and I was much pleased by how easily it allowed us to think abstractly about information relationships. Zope provides a simple object model for web publishing; in effect, it transforms web publishing from a process of *retrieving files* or producing *database queries* to one of *publishing objects*—a shift to a considerably more sophisticated level. See http://www.zope.org

modern notebook computers—impressed me immensely. Kay's 1996 history outlines the practical emergence of the "object paradigm" amid his research project to develop personal computing for generations of children to come. It is, in a sense, a *post-hoc* manifesto for an educational and technological vision. I was coming, by degrees, to appreciate this vision and its many connecting points to pieces of my own history. On the strength of this encounter, I then began to recognize the various themes that had made an impression on me—HyperCard, MOO, groves, learning objects, Zope—as variations on and deviations from this core tradition; that here was the original source of this tradition of thinking about computing, media, and information design. What impressed me most was that this tradition *began with an educational vision*: the original beneficiaries of this thinking were children, and it was only considerably later that personal computing and object-oriented programming became wrapped up with business needs and office productivity. The more I began to look at the history of computing over the past three decades, the more I realized that the tradition represented here—centering on Alan Kay's multi-decade project—had enormous importance to the study of computing in education, far more importance than is commonly appreciated. I had found my thesis topic.

Why This Study? Why This Approach?

Reflecting on my history

When I returned to school in 1999 with an agenda to study technology from an educational perspective, it was with a particular set of contraints that I began to set the scope of my work.

- I saw a real need for any work I was to undertake—be it development or analysis—to have some sense of historical embeddedness. I had learned this during my work on the SGML project at Open Learning Agency, and I felt that for my work at the doctoral level, it was essential that awareness, if not respect, for what had gone before had to be a foundational piece.

- I had developed a jaded attitude toward ambitious projects and bold, 'new' ideas; having personally been through the belly of a few of these, I had the sense that a good proportion of the energy that drives innovative projects comes from the promise of proving something (or oneself) to naysayers. Every successful endeavour is a rally against cynicism, but where this itself becomes the driver, healthy enthusiasm gives way to hubris.
- I had come to recognize the "microclimate" surrounding most educational technology projects, within which all things are possible and kids do "wonderful" things. Beyond the fragile boundaries of the sphere of energy provided by a few inspired individuals (often teachers), duplication or scaling-up of such projects is impossible. I later ran into this concept better articulated as the "miracle-worker discourse" (de Castell, Bryson, & Jenson 2002).
- I had the sense that the division of labour between experts and end-users was pathologically implicated in most educational technology projects, and that somehow this reification needed to be resisted.
- I was wary of—if not positively hostile to—baldly technocentric thinking; there was no way I was about to uncritically engage in any actual development projects; I returned to school determined to not be anyone's "webmaster," nor to work on any "e-learning" projects, despite numerous opportunities to do so. Part of my intent in returning to school was to claw my way back from my position on the 'techie' side of the division of labour. For, lest we ascribe too much power to the (vaguely defined) 'technocrat,' we should remember that technical work is still *work*—that it is labour, carried out according to the same kinds of logic that governs labour in more traditional contexts. Technical work carried out in a naïve technocentric mode is in the worst cases straightforwardly exploitative, in the best cases still narcissistic.

"Computer criticism"

In a 1987 article in *Educational Researcher*, Seymour Papert proposed a genre of writing he called "computer criticism." I felt that my work should take this to heart. But what would "criticism" mean, exactly, in the context of educational technology? Contrary to the wealth of disengaged condemnation that bills itself as critique, criticism in the larger sense properly demands a certain kind of engagement to be meaningful. As Papert pointed out:

> The name does not imply that such writing would condemn computers any more than literary criticism condemns literature or social criticism condemns society. The purpose of computer criticism is not to condemn but to understand, to explicate, to place in perspective. Of course, understanding does not exclude harsh (perhaps even captious) judgement. The result of understanding may well be to debunk. But critical judgment may also open our eyes to previously unnoticed virtue. (Papert 1987, p. 22)

I have seen very little writing on educational technology that lives up to what Papert proposed here.[10] Criticism in the sense described here requires not just a familiarity but a fluidity and fluency with the issues, the discourses, and the practices of computing. It requires a sense of and appreciation of where these discourses and practices come from, historically and topologically—that is, beyond the disciplinary boundaries in which they may immediately be seen. In a very important sense, Papert's computer criticism *requires* the breaking down of or at least resistance to the division of labour which sets technology and human aims apart. Rather, in order to make sense, we have to be able to see how these positions relate to and give rise to one another.

Multiple perspectives, blurred genres

This, then, has been my starting place: my highest-order goal in this process is to attack or subvert the taken-for-granted division of labour that inscribes the boundaries of technology and which makes Papert's "computer criticism" next to impossible. Whatever potential for empowerment and democracy may come with information technology is repeatedly under-

10. Papert's own students are, perhaps predictably, exemplary: e.g., Bruckman 1997; Goldman-Segall 1998; diSessa 2000.

mined by the reification of these positions: experts and engineers vs. end-users and consumers. That alternatives are possible—and indeed extant—is the case which I seek to make here. To this task I bring my own background; as a student of education and culture, and also as a designer, creator, and teacher on the technical side. In what follows, I am trying to deliberately blur the lines between computing, social science, cultural criticism, and education. I am, to use Latour's (1993) vocabulary, interested in actively proliferating the hybrids.

Of course, this isn't a matter of so much conscious choice on my part. My personal exploration and growth in understanding of all sides of these issues is itself varied and begins from multiple perspectives. Because of this I cannot choose sides or loyalties in the division of labour; I stand with feet on both sides. And because of this, because of my positionalities, I am faced with real methodological constraints: I cannot honestly bracket out the issue of expertise, hide in the inscription of an "outsider" analyst, nor pretend to naïveté, as some have attempted in the sociology of science. I have come to a place where I can inhabit neither the "miracle worker" role nor that of the "humanist" critic—for these are caricatures which are products of the very division I seek to break down. And so I have undertaken to do this research, and to write this work, from multiple active perspectives: from the standpoint of educational research and cultural criticism and also, simultaneously, from a standpoint of some technical sophistication, to walk the in-between space between the romantic and the ironic: to be more, and to write more, than caricatures.

My challenge, then, is twofold: in the first place, my task here is not one of building—nor singing the praises of (that is, *instructing*)—another tool or toolset. I must aspire to a 'higher' analytic frame, as the literary critic does compared with that of the author. In the second place, however, intellectual honesty demands that I not restrict my stance to that of observer; in order to meaningfully reflect on what I am studying, I must bring to bear my full faculties, including my experience and positionality—my own *history*—as a designer and builder of systems. At a theoretical level, this double role is of course the case in *literary* criticism; a critic is of course also an author—of criticism if not of 'literature' *per se*; how could it

be otherwise? In a practical sense, though, this doubling is problematic in the case of technology—of "computer criticism"—because most of us do find ourselves on one side of the divide or other: designer or user? Do these two exclusives represent our full realm of possibility?

In order to do effective *criticism*, in Papert's sense, I have to transcend that divide—to strike a balance between criticism and advocacy. I have to find a line (or invent one, or at least spend some time mapping one out) between philosophizing about educational IT "from 30,000 ft"—that is, from a disengaged, 'outsider' perspective—and uncritically evangelizing a particular technology or technological vision from the inside.

Methodology and the Problem of Distance

The study presented here is a historical one. My intent is to present the story of the Dynabook project and to trace its historical trajectories. My challenge, thus, is to write history effectively, and to write effective history; as this is a cultural historical account, my methodological concerns must also bear upon traditions of making sense of culture.

As a former student of anthropology, my unsurprising point of departure for the "interpretation of cultures" is the work of Clifford Geertz, who had ethnography in mind when he wrote his groundbreaking essays on the turn to a interpretive—hermeneutic—approach to the study of culture. Geertz' admonition that the analysis of culture is "*not an experimental science in search of law but an interpretive one in search of meaning*" (1973, p. 5) is as applicable to the business of historiography as it is to ethnography, as a body of work emphasizing *cultural history* shows. It is worthwhile tracing the "interpretive turn" backwards from Geertz to French philosopher Paul Ricoeur, whose article "The Model of the Text" (1971/1991*a*) set the stage for the adoption of interpretive 'method' in the social sciences—particularly in the English-speaking world—in the 1970s. Tracing back further from Ricoeur takes us to the major source of modern philosophical hermeneutics in the work of German philosopher Hans-Georg Gadamer.

Gadamer's argument in his magnum opus, *Truth and Method* (1975/1999), is that it is *only* through participatory knowledge and interpretation that we are able to come to *any* kind of an appreciation or critique of what we study. This is significantly at odds with the more traditional methodological quest for objectivity, and the tension here has been a central issue in social sciences for the past half-century; it puts in question the very idea of social "science" and is the crux of Gadamer's questioning of "method." The task of researching human life and *meaning* cannot possibly proceed according to the objectivist ideals of natural science, since the question of meaning can only be approached by engaging with the subject at hand, intersubjectively, participatory, performatively. Gadamer wrote, "The concept of the life-world is the antithesis of all objectivism. It is an essentially historical concept, which does not refer to a universe of being, to an 'existent world'" (1975/1999, p. 247). Jürgen Habermas is perhaps most succinct in his explication of this concept:

> Meanings—whether embodied in actions, institutions, products of labor, words, networks of cooperation, or documents—can be made accessible only *from the inside*. Symbolically prestructured reality forms a universe that is hermetically sealed to the view of observers incapable of communicating; that is, it would have to remain incomprehensible to them. The lifeworld is open only to subjects who make use of their competence to speak and act. (Habermas 1984, p. 112)

That the lifeworld is essentially historical makes a particular demand on the inquirer. The challenge is not to make sense of social or cultural phenomena *qua* phenomena; that is, awash in a vast and complex present tense, in some vain effort to discern the structures there. Rather, it is to make sense—make meaning—by engaging with the historicity of the things and ideas and patterns we seek to understand. It makes the writing of history—no less than the writing of ethnography—necessarily and inescapably reflexive. Gadamer's powerful vision of this process is centered around the concept of *tradition* (1975/1999 p. 284ff) which, following Heidegger's philosophy, makes the temporal unfolding of experience paramount. Richard Bernstein summarizes:

> As Gadamer sees it, we belong to a tradition before it belongs to us: tradition, through its sedimentations, has a power which is constantly determining what we are in the process of becoming. We are *always already* "thrown" into a tradition. We can see how far Gadamer is from any naïve form of relativism that fails to appreciate how we are shaped by effective history (*Wirkungsgechichte*). It is not just that works of art, text, and traditions have effects and leave traces. Rather, what we are, whether we are explicitly aware of it or not, is always being influenced by tradition, even when we think we are most free of it. Again, it is important to reiterate that a tradition is not something "naturelike," something "given" that stands over against us. It is always "part of us" and works through its effective-history. (Bernstein 1983, p. 142)

The notion of effective history—and, as Gadamer extends it in *Truth and Method*, "historically effected consciousness"—implies the kind of engagedness that binds me as a researcher, and in which I eschew methdological distancing. In fact, the opposite seems to be the rule: "Understanding is to be thought of less as a subjective act than as participating in an event of tradition, a process of transmission in which past and present are constantly mediated" (Gadamer 1975/1999, p. 290).

Engagement with and participation in historically embedded situations—that is, within traditions—implies *making judgments* about what is good, what is valid, what is just. This constructive engagement is used in Goldman-Segall (1998) as a methodological tenet in her strategically decentered ethnography. It is the basis of Rudolf Makreel's (2004) notion of "ethically responsive" history. Philosopher Alasdair MacIntyre (1984) takes this thread from a history of philosophy into a theory of ethics. Jürgen Habermas (1984) takes it to a theory of communication. In all of these variations, there is no appeal to external authority or source of rationality but the tradition itself, in which the researcher is necessarily—partially at least—*embedded*. Habermas writes:

> The interpreter would not have understood what a "reason" is if he did not reconstruct it with its claim to provide grounds; that is, if he did not give it a *rational interpretation* in Max Weber's sense. The *description* of reasons demands *eo ipso* an *evaluation*, even when the one providing the description feels that he is not at the moment in a position to judge their soundness. One

> can understand reasons only to the extent that one understands *why* they are or are not sound, or why in a given case a decision as to whether reasons are good or bad is not (yet) possible. An interpreter cannot, therefore, interpret expressions connected through criticizable validity claims with a potential of reasons (and thus represent knowledge) without taking a position on them. And he cannot take a position without applying his *own* standards of judgment, at any rate standards that he has made his own. (Habermas 1984, p. 116)

I see the basic question, or problematic, in historiography to be the distance between the storying of the historical subject and the storying of the historian; the distinction between primary and secondary sources is necessarily blurred. I do not take this to be a *problem* of validity—nor of endless layers of relativism—rather, in line with the hermeneutical approach, I see this straightforwardly as the space of interpretation. On the question of the relationship between lived experience and narrated story, I do not see a "discontinuity" between these, as Hayden White (1973) has famously claimed; rather, I believe, with Gadamer and his followers—notably Alisdair MacIntyre (1984), David Carr (1998), and Andrew Norman (1998)—that lives are lived and *made sense of*, in the first person, narratively. The conception of the unfolding of experience, and the hermeneutic circle of its ongoing interpretation in time is grounded in the Heideggerian phenomenological tradition, and I take as a basic methodological principle the treatment of experience, and action, as *text* (Ricoeur 1971/1991*a*). The historiographical challenge, then, is not one of gaining privileged access to or an unadulterated representation of what has gone before, but rather one of entering into a meaningful engagement with the myriad layers of effective history which themselves produce the possibility of such an engagement. This is why the details of my personal history with respect to computing and computing cultures are important to this account, as is my process of assembly and immersion in the sources I bring to bear here. It is worth noting in passing that nothing in this framing would be out of place in a contemporary exposition on ethnographic method, but the present study is not ethnography. Rather, I present here an analysis of a body of historical documents—to discern and docu-

ment the traditions, genres, and apparent structures of discourse and practice. I will attempt to outline the details of this approach presently.

Introducing genre theory

Drawing on the historiography of Mark Salber Phillips, and contrary to Hayden White's somewhat monolithic and ahistorical notion of *emplotment*, it makes sense to speak of *historically contingent genres* of lived narrative. Phillips argues that history is best seen as a family of historically contingent and often overlapping genres. This approach—building on White's opening of historiography to literary theory and concepts—points to a rich interdependence of text, context, readers, and writers (Phillips 2000, p. 10). "Genre, of course, is not a self-contained system. It is a way of ordering and mediating experience, literary and extraliterary" (p. 11).

Genre theory—as it appears in historiography (Phillips 2000), in literary theory (Jauss 1982; Cohen 1986; Miller 1994*a*; 1994*b*), in linguistics and cognitive science (Swales 1990; Scollon 1992; Bazerman 1998; Lemke 2001)—opens up the possibility of considering the material in question directly in the context of the communities (scholars, audiences, markets) of people for whom the material is relevant. Swales is perhaps the most direct: "genres are the properties of discourse communities" (Swales 1990, p. 9). This works bi-directionally in a sense; genres are defined by the communicative acts of discourse communities, who are, in turn, constituted by communicative acts understandable within certain genres. For this to work, we must ensure that we do not reify genres as sets of analytic characteristics or properties (Cohen 1986, p. 210). Bazerman puts it most eloquently:

> By genre I do not just mean the formal characteristics that one must observe to be formally recognized as correctly following the visible rules and expectations. Genre more fundamentally is a kind of activity to be carried out in a recognizable textual space. [...] Thus genre presents an opportunity space for realising certain kinds of activities, meanings, and relations. Genre exists only in the recognition and deployment of typicality by readers and writers—it is the recognizable shape by which participation is enacted and understood. (Bazerman 1998, p. 24)

In putting the emphasis on genres as structures of activity and *participation*—rather than just of literary forms or bloodless "discourse communities"—Bazerman returns us to the continuity of narrative as lived and narrative as written: that the modes of interpretation of one's life and experience "typically" fall into various historically and culturally contingent patterns or categories (cf. a very similar framing in MacIntyre 1984, p. 212). Thus there is no essential difference between the treatment and interpretation of one's lived narrative and that of the subsequent "historical" narrative; both are literary/interpretive acts, both operate according to conventional, historically conditioned genres and modes. Genres, then, are "perspectivity technologies" (Goldman & Maxwell 2003) that mediate between individual meaning and social practice. What this means for the present study is that genres are among the fundamental units of analysis, and, following Phillips (2000), my intent is to focus on the rise and fall of particular genres in historical context. In this sense, the subject of this study is the *emergence*, *translation*, and relative *survival* of a set of genres of computing and technocultural discourse.

History as politics

The historian's task is "not merely a reproductive but always a productive activity as well" (Gadamer 1975/1999, p. 296). What makes the writing (and reading) of history interesting and useful is the dynamic between these interpretive locations. The writing of history is thus not an attempt to nail down what happened in the past, nor to explain the present by the past, but rather an attempt to "shape and re-shape our collective understanding" (Norman 1998, p. 162) of the past, the present, and, more importantly, the future. The writing of history is therefore a generative, engaged, political process that is itself historically embedded. This very embeddedness, and not any methodological measure, is what keeps us from hopeless relativism. I am not—nor is any writer—in the business of telling *a story* in the sense of *one story* among many; I can only write *the* story *according to me*, from my particular perspective. What I must strive to do—and here is the role of methodology, discipline, and rigour—is increase where possible the number and variety of perspectives

considered and repeatedly test the story against existing frameworks; that is to say, to maximize the *connectedness* of my account. Lather's (1991) discussion of triangulation, construct-, face-, and catalytic-validity guidelines are directly instructive here, and serve the same overall end of connectedness. It is only possible for me, as an embedded subject, to tell one story—the story I am *given*, shaped by my historical and hermeneutical *horizons*—the story which emerges for me in the process of my investigation. I can speak of "a story" among many possible stories, but I can only honestly mean this plurality as a reference to the possibility of or invitation to other stories, which can only be evoked or juxtaposed, and never directly represented—to do so would require that I actually have some means of getting outside the stories, to live outside the lifeworld, outside of narrative. *The story*, then, is necessarily incomplete, its only hope for greater completeness is to be futher embedded.

The Dynabook as/in history

> *Studying a technological project isn't any harder than doing literary criticism.*
>
> – Bruno Latour, *Aramis*

To frame my process and thus the scope of the story I will tell, a brief overview of sources and my engagement with them is in order. My starting point for this research is a rich reflective document by Alan Kay dating from the early 1990s, "The Early History of Smalltalk," prepared as a conference presentation in 1993 for the ACM's *History of Programming Languages II* and subsequently published (88 pages worth) by the ACM[11] in a large volume compiling formal articles on the sessions with appendices, presentation transcripts and discussants' remarks (Kay 1996*a*). Kay's piece serves as a bird's-eye view of the first decade or so of his work, as well as a hub document for subsequent research. "The Early History of Smalltalk" is still referenced and recommended within the development community as probably the single best articulation of Kay's vision and work, especially in the 1970s.

11. The ACM—Association for Computing Machinery—is computing's pre-eminent professional association. It operates a press and an enormous library (digital and otherwise) of published material, proceedings, and literally dozens of periodical publications running the gamut from newsletters to scholarly journals. In the world of scholarly communication, the ACM is an exemplar, and is certainly a terrific resource to the historian.

Moving out from this point, I located two main bodies of literature which, though grouped chronologically, I distinguish contexually rather than as primary vs. secondary sources. The first, dating mostly from the 1970s, are conference papers and research reports from Xerox PARC, outlining the Dynabook vision, the details of the early Smalltalk implementations, and a smaller amount of material on educational research. In this literature, there are a small number of documents from the early 1970s, and then nothing until 1976, apparently due to a Xerox crackdown on publicity in the wake of a revealing article by Stewart Brand in *Rolling Stone* magazine in 1972—an article which portrayed the PARC researchers as long-haired "hot-rodders" (Brand 1972) and about which Xerox executives were somewhat embarrassed (Kay 1996*a*, p. 536). After 1976 there is a wealth of material documenting research and development; this literature culminates in the early 1980s with the end of the original Xerox PARC teams and the release of the Smalltalk technology to the world beyond Xerox.

A second body of literature I have identified follows on Kay's 1993/1996 publication, and comprises a large number of journalistic articles and interviews with Kay as well as a substantial number of lectures and presentations by Kay (many of which, thanks to the Internet, are available online as digital video, audio, or text transcripts). This second body of literature roughly coincides with the public release of the Squeak technology by Kay's team at Apple Computer and, later, Disney Corporation in the 1990s. The Squeak technology was positioned as the re-realization of portions of the original Dynabook vision, and not surprisingly, the literature appearing in the late 1990s and early 2000s is concerned largely with the re-articulation of that vision and its key principles, often in much more detail than the works from the 1970s, but certainly in line with them conceptually.

Given these two groupings—material from the Xerox PARC period (1971–1983) and then from the mid 1990s onward—there is clearly about a decade in which very little published material appeared. During most of this time, Alan Kay was at Apple Computer and working on educational research with a very low public profile; I have had to do some digging to turn up a number of unpublished reports from this period[12]—a phase I think is

critical to an appreciation of the project's re-emergence (with Squeak) in the late 1990s, ostensibly a continuation of the original 1970s vision, but now appearing in a very different world.

A third "body of literature" I encountered is the archived Internet communications of the project teams and worldwide community surrounding the Squeak project since the 1990s; this comprises several mailing lists, a dozen or so websites, and the archive of the various versions of the software released on the Internet since then. A number of instructional and/or reference books on Squeak have been published since the late 1990s as well, and I include these in this latter category.

Having oriented myself to these three major document groups, it then became possible to interpret a large body of surrounding literature through the lenses of the Dynabook vision and its various research projects; this surrounding literature includes the documentation for the Smalltalk language (aimed at software developers rather than children or educators) from the 1980s; a wide variety of educational research and development work spanning three decades or more from MIT's Media Lab, with which Kay has been loosely associated over the years and which offers something of a parallel research agenda, particularly in the work of Seymour Papert and his successors; and, more widely, a large body of literature on educational computing, computer science, and Internet culture over the past two or three decades.

My trajectory took me through much of this documentary material, and having absorbed and made sense of a good deal of it, at least on a superficial level, I made personal contact with several of the key individuals in this story. In the spring of 2004 I travelled to Glendale, California, to spend two days at the Viewpoints Research Institute with Alan Kay and Kim Rose, during which time serendipity allowed me to meet and talk to both Seymour Papert and Smalltalk developer Dan Ingalls. In the fall of that year, the *OOPSLA '04* confer-

12. Alan Kay and Kim Rose at the Viewpoints Research Institute are to thank here for opening their substantial archives to me; Ann Marion, project manager with Kay while at Apple Computer, similarly deserves thanks for sharing her own archives. But while my research of this period serves to capture the main currents of research and thinking, I make no claim to have exhaustively covered this period; there remain mountains of primary documents from the 1980s that I did not cover, including hundreds or thousands of hours of video.

ence happened to be in Vancouver, which brough Kay and a good number of the Squeak community to my own hometown, and which provided an excellent opportunity to talk with many of these people—in particular, I had opportunity to interview Ted Kaehler, who had been part of Kay's team since the early 1970s, and whose exceedingly detailed and organized memory of the past thirty-five years proved invaluable in answering myriad detail questions in my construction of this narrative. In a similar vein, conversations with Kim Rose and Ann Marion—both of whose relationships with the project date to the mid 1980s—filled in many gaps in my understanding.

I want to point out, though, that I do not consider this to have been an interview-driven research project; my primary thread has been the consideration and interpretation of written texts, and I see these many valuable conversations as supporting that primary documentary work. I spent the better part of two days talking with Alan Kay while I was in Glendale, but there was very little of that wide-ranging and sublimely tangential conversation that would be recognizable as 'interview'—a judgment Kay notably approved of.

In retrospect at least, I feel there is something of the spirit of Jerome Bruner's "spiral curriculum" in my research method—a 'developmental' re-framing of the hermeneutic circle. I have gone around and around the literature, reading, re-reading, and adding to the collection as I have gone along; at each re-consideration and re-interpretation of my sources, and certainly through the process of writing and re-writing this account, I have achieved what I hope are both deeper and more broadly connected interpretations and framings of the various facets of the story. I consider this to be an example of how hermeneutically informed historical inquiry ought to move: an iterative process of the "fusing of horizons," of achieving a particular instantiation of the "unity" (as Gadamer would have it) of self and other. In the depth of time and interest I have engaged with it, it is hopefully rich and at least defensible in terms of its "validity." It is also entirely incomplete, inexhaustive, and open to debate. Whether this study succeeds I believe should be judged in terms of the value and appeal of such debates.

How Do We Know a Good Idea When We See One?

One of Donna Haraway's great themes is that none of us is *innocent* in the realm of techno-science. It is only by virtue of our involvement and investment with these issues that we are able to make any *sense*, or make any *difference*. And so we can only attempt to own and own up to our position(s). Haraway writes:

> The point is to make a difference in the world, to cast our lot for some ways of life and not others. To do that, one must be in the action, be finite and dirty, not transcendent and clean. Knowledge-making technologies, including crafting subject positions and ways of inhabiting such positions, must be made relentlessly visible and open to critical intervention. (1997, p. 36)

I bring this point to the foreground in order to add one more layer of context to the present study: to forswear the objective of critical distance. To write an "agnostic" study of a case like the one I an writing here—the historical trajectory of Alan Kay's Dynabook vision—would be to limit the significance of the study to the 'conclusion' that a particular technological project failed, or succeeded, or enjoyed market success, or withered away. It would not, on the contrary, be able to speak to the ethical dimension of such a trajectory and to suggest why we might care. This is to say, given the ambitions of this study as I have framed it for myself, it would not be possible to do the present inquiry while holding to an ideal of objectivity or agnosticism. For the overarching aim of this study—and I take "aims" to be foundational in deciding methodological issues—is to break down the division of labour that we take for granted, to subvert the reified discourses of 'experts,' 'engineers,' 'designers,' 'end-users,' 'miracle workers,' 'plain folks' and so on, an aim I think is possible at least by making the different threads and voices herein aware of one another: by building bridges, or at least by drawing attention to the always-already constructed boundaries inscribing our various relations to technoculture. I want the readers of my work to be able to read these discourses, evaluate these practices, with a richer, more critical eye than the usual rhetoric surrounding technology (especially in education) affords. This is *precisely the goal* of criti-

cism in the large; it provides us with improved means for making practical judgements, for identifying the right thing, or the good thing—or at least the "better" thing.

In order to answer the question, "how do we recognize a good idea when we see one?" we have to first recognize a good idea. This cannot be done while maintaining a stance of agnosticism or methodological distance. It can only be accomplished by engagement and then elaboration of the layers and spheres of meaning which can be generated within that space of engagement—it demands, as Haraway says, being "finite and dirty."

Personal computing/educational technology as a site of struggle

> *Thus what essentialism conceives as an ontological split between technology and meaning, I conceive as a terrain of struggle between different actors differently engaged with technology and meaning.*
>
> – Andrew Feenberg, *Questioning Technology*

Philosopher of technology Andrew Feenberg's conception is a response to the burden of essentialism, inherited from centuries of European philosophy. Feenberg's stance—*technology as site of struggle*—is a starting point for me. I conceive of technology, information technology, educational technology, not as a thing or a discourse or a set of practices to be analyzed or judged, but as a contested ground, over which will be fought the battles concerning democracy, education, and capitalism in this century.

Personal computing is "socially constructed"—it is facile to point this out. In its construction is found the site of a struggle for meaning and significance. The substructure of this struggle is in the competing genres of understanding and articulation of what personal computing—and educational computing—are about. My starting assertion is then that the Dynabook is a volley thrown into the midst of that struggle. It is not purely an 'historical' artifact whose day was in the 1970s; rather, the thrust of the Dynabook project is at least as relevant today in the 21st century as it was three decades ago. This is a battle for meaning, a battle to define what computing is and shall be, and it is far from decided. The stakes of this battle are, I believe, higher than we tend to admit, as our default instrumentalist stance leads us to downplay the significance of this struggle. How computing is defined

in these early years (in its *incunabula,* Kay would tell us) will have enormous consequence for how we live and work and learn and conduct our communities in the future. I mean this in the most concrete sense, but I am not interested here in evaluating any particular educational technology in terms of *instructional* efficacy. Rather, my interest is at a broader level of citizenship and democracy, in the sense that John Dewey established:

> A democracy is more than a form of government; it is primarily a mode of associated living, of conjoint communicated experience. the extension in space of the number of invididuals who participate in an interest so that each has to refer his own action to that of others, and to consider the action of others to give point and direction to his own... (Dewey 1916, p. 86)

This may sound like an unabashedly Romantic framing of the issue—Kay has acknowledged as much on numerous occasions, and so should I. I am not neutral; I am not here to play the disengaged observer. I am interested in casting my lot for some genres and not others: genres which define and structure practice, interpretation, understanding as political and ethical patterns. Historian Mark Salber Phillips studied the rise and fall of genres of historical writing/understanding in 18th-century England. I see the present project as doing something related for the technocultural genres of the past three decades.

My task, then, is to demonstrate how and why technology is political—more specifically, *software as politics by other means*. There is a superficial interpretation of this identification which concerns the way in which software is designed, released, and licensed in ways which to a greater or lesser extent constrain and direct a "user's" actions and potential (Lessig 1999; Rose 2003); this is an arena of ongoing battles in copyright law, emerging trends like free and open-source software, the reach of open standards, and the market dynamics of dominant corporations such as Microsoft, Adobe, Apple, and others.

But a second and somewhat deeper interpretation of technology and software as politics has to do with a gradual (but by no means consistent nor even particularly widespread) democratization of computing over the past four decades. In this interpretation, it becomes possible to identify and map out the interplay of power and resistance within a technical

sphere like the Internet or even your desktop. This is the arena in which much of Alan Kay's project can be considered. This is the level at which the educational implications of computing are most acute, and, indeed, most accessible, if we take the time to look.

A third, more systemic interpretation requires a foray into philosophical theories of technology, locating the political aspects of technology at their lowest and most general level. It is to this topic that I turn next.

Chapter 3: Framing Technology

In order to do justice to a treatment of technological development, I want to first establish a philosophical position with respect to technology. In doing so, my aim is to set up a theoretical scaffolding upon which I can hang particular reflections in my examination of Alan Kay's vision of personal computing. My intent here is not to provide a broad-strokes "theory of technology"—neither by means of reviewing a hundred years' worth of philosophizing on the topic nor by attempting to construct a water-tight model. What I will try to present here is a provisional framing that draws attention to a number of particularly interesting characteristics of technology.

The framing I have in mind rests on a foundational concept of technology as *media* and as *mediation*. Building on this, I will introduce the "sociology of translation" as advanced in the work of Bruno Latour and Michel Callon, and I will elaborate the implications of *translation* as a principal metaphor for the dynamics of technocultural systems. This semiotic aspect of technology leads to a discussion of what I have called the "mechanics of text," and here I wish to present a latter-day framing of the kind of media ecology Marshall McLuhan outlined in *The Gutenberg Galaxy*, but with digital software given the central role, rather than print. The emphasis on software itself leads to a consideration of *simulation* as the paradigmatic practice of dynamic digital media, and I will argue that this is an essentially hermeneutic process, something which should focus our critical attention on its *situatedness*. Finally, I will discuss some methodological implications raised by this framing of technology, and put the spotlight on the ethical and political considerations of a cultural treatment of technology.

Technology as Media

My starting point is to treat technology as *mediation*—or as *media,* this word bringing with it a particular set of connotations and references (largely set in their current form by McLuhan in the 1960s). To treat technology as media is to establish a perspective probably

distinct to the "information age." When the artifacts provoking discussion are the Internet and computing devices, a decidedly different tone is set than with industrial technology such as steel mills and power stations. This distinction immediately brings forth the *hermeneutic* aspect of technology (Feenberg 1999; 2004), by which I mean the complex of forces and processes which govern the significance of particular technologies to particular people in particular times and places. And it is the hermeneutic question that I mean to foreground here, rather than any suggestion of an *essence* of technology.

> In sum, differences in the way social groups interpret and use technical objects are not merely extrinsic but also make a difference in the nature of the objects themselves. *What* the object *is* for the groups that ultimately decide its fate determines what it *becomes* as it is redesigned and improved over time. If this is true, then we can only understand technological development by studying the sociopolitical situation of the various groups involved in it.
>
> (Feenberg 2004, p. 216)

A treatment of technology as media or mediation lends itself also to the exploration of a *media ecology*. Now, by ecology I do not mean anything *green* (at least not directly). Rather, what I mean by ecology is, after Postman (1970), a dynamic, evolving *system* in which actor and environment are inseparable and mutually constitutive, in which both people and cultural artifacts are considered, and in which responsibilities and ethics are emergent and situated, in which the content-context distinction itself is problematic and should probably be avoided, for everything is the context for everything else. But, lest I risk characterising the whole issue as a sea of *aporia*, let me appeal back to mediation and hermeneutics—which provides a vocabulary for dealing precisely with this sort of contextualism.

From a hermeneutic perspective, we *are* our mediations; mediation is primary, and not something that happens to already-existing entities. What this implies is that, as in McLuhan's famous aphorism, *the medium is the message* (and that practically speaking, it is an error to attempt to distill one from the other), we human beings are effectively inseparable from our technology/material culture. Is this so radical a stance? It would not be terribly controversial to claim that human beings are inseparable from our language(s). I mean to

claim for technological mediation this same primacy, to position *technology as language*, or at least to suggest that it be treated similarly. I make this claim not as part of a characterization of modernity, but as a fundamental part of what being human is about.

Putting mediation first dissolves the debate between the idea that language expresses pre-existing thoughts and the notion that we are trapped within the limits of our language; or, similarly, whether culture is something internal or external to individuals. Michael Cole's influential book, *Cultural Psychology: A Once and Future Discipline* (1996) draws upon the Russian cultural-historical school (after Vygotsky) to elaborate a theory of mediated, contextualized action which "asserts the primal unity of the material and the symbolic in human cognition" (p. 118). In Cole's version of mediated action,

> artifacts are simultaneously *ideal* (conceptual) and *material*. They are ideal in that their material form has been shaped by their participation in the interactions of which they were previously a part and which they mediate in the present. ... Defined in this manner, the properties of artifacts apply with equal force whether one is considering language or the more usually noted forms of artifacts such as tables and knives which constitute material culture. (p. 117)

If we *are our mediations*, then certainly we cannot posit that ideas precede expression or mediation; nor can we accept that we can only think what our language of expression makes possible, for we can invent languages (and indeed do). So with our technologies: we do not exist in some essential way prior to technological mediation, nor are we subsumed within a technological trajectory. We invent tools, as we do languages, and subsequently our experience and agency is shaped by them. But to say this is merely to ape McLuhan; it is the historicity of these 'inventions' that is the particularly interesting story.

Precisely in the dynamics of this relationship and how 'we' (I will put that in qualifying quotes this time) see it—in terms of historicity, class, gender, politics, and so on—are found the interesting and important stories; that which is most worthy of study and the investment that leads to deeper understanding. Far from seeing technology as something that augments or diminishes—or indeed qualifies—the human, my starting place is that human-

ity is itself technologically defined, and in myriad ways.[1] Donna Haraway's *cyborg* trope is a particularly eloquent and evocative address to this notion:

> There are several consquences to taking the imagery of cyborgs as other than our enemies. [...] The machine is not an it to be animated, worshipped, and dominated. The machine is us, our processes, an aspect of our embodiment. We can be responsible for machines; they do not dominate or threaten us. (Haraway 1991, p. 179)

There is a decidedly historical character to such framings and perspectives. How this relation presents itself to us today in the age of the "cyborg" is not what it would have been in the "machine age" of steam and steel; nor would it have the same character in the 13th century, with its 'machinery' of iconography, horsecraft, and emerging bureaucracy. But what is the same in all these cases is the central role of technological mediation.

What is mediation, then, or media? I want to spend some time teasing out the implications of these concepts by going down two different—but complementary—routes. The first route is the work of architect Malcolm McCullough; the second is by way of the sociology of technoscience of Bruno Latour and Michel Callon.

McCullough's Framing of Media

To turn to the more specific formulation, then, what is a *medium*? To my mind, nobody answers this better—and in a definitively active, constructive, and contextualist mode—than Harvard architecture professor Malcolm McCullough, in his 1998 book *Abstracting Craft*. McCullough writes:

> Tools are means for working a medium. A particular tool may indeed be the only way to work a particular medium, and it may only be for working that medium. Thus a medium is likely to distinquish a particular class of tools. [...] Sometimes a medium implies such a unique set of tools that the whole is referred to without differentiation. Painting is a medium, but it is also the use of specific tools and the resulting artifact: a painting. The artifact, more than

1. Bruno Latour's *Pandora's Hope* (1999, pp. 202–213) traces a possible history of technocultural mediation which begins even before the primordial "tool kit" of stones and sticks: with the very idea of social organization—as technology.

> the medium in which or tools by which it is produced, becomes the object of our work. [...] Artifact, tool, and medium are just different ways of focusing our attention on the process of giving form...
>
> In many refined practices, the perception of a medium surpasses any perception of tools. If a medium is a realm of possibilities for a set of tools, then any immediate awareness of the tools may become subsidiary to a more abstract awareness of the medium. (McCullough 1998, pp. 62–63)

McCullough positions media in the midst of the "process of giving form"—that is, *practice*. His framing integrates material and conceptual resources equally, suggesting that these become distinct as we focus our attention differently. McCullough clearly echoes Heidegger's famous ontology of the hammer as *ready-to-hand*. He also, as we will see, parallels Latour's vocabulary of articulation and network; the hand and tool and medium become a network, aligned in the articulation of the work. McCullough's exploration of the subtleties of craft—and particularly handcraft—moves from tool to medium to artifact seamlessly, drawing his attention to the spaces of tension and grain within each, letting his attention fall away where it is not needed, according to the dynamics of actual, situated practice.

Note the emphasis in McCullough's account on *work*, and in particular form-giving work. This is a participatory stance, not a spectatorial one; it bases the ontology of media in the actor, and not in the spectator or consumer. Compare McCullough's framing with that of Peter Lyman, writing on the computerization of academia:

> [M]ost fundamentally, most people only want to 'use' tools and not to think about them; to nonexperts, thinking about tools is a distraction from the problem presented by the content of the work [...] Whereas a machine has a purpose built into its mechanism, and a tool requires the novice to acquire skill to realize this purpose, a computer is a field of play only to an expert. (Lyman 1995, p. 27)

Lyman's analysis makes 'tools' into rather ahistorical black boxes while appealing to higher-order goals—which may prove to be something of a false economy. How a particular tool comes to be neatly integrated in a particular practice—to the point where it becomes subsumed into the practice—is a profoundly historical and political process (Franklin

1999). This notion is often troublesome, but there exists a case in which nearly everyone recognizes the historicity of 'practice': the relationship between a musician and her instrument. Both McCullough and Lyman touch on musicianship, but treat it rather differently. Lyman goes to far as to give special status to this case, claiming that the musician/instrument connection transcends tool use: "In performance, the musical instrument and player interact in a manner that cannot accurately be described as a human-tool relation" (Lyman 1995, p. 29).

McCullough's treatment is much more *involved*: "acute knowledge of a medium's structure comes not by theory but through involvement" (McCullough 1998, p. 196). This awareness or knowledge has two faces: one is the familiar falling away of intermediaries to allow consciousness of the medium or practice itself. In the other are the traces of culture, knowledge, and history that become wrapped up in our tools, media, and artifacts. The guitar may disappear from the consciousness of the musician, such that she becomes aware only of the music, but over time, the instrument will bear the marks of her playing, will show wear from her hands, be stained by the moisture from her skin; conversely, her hands and her musicianship in general will bear the complementary wear patterns. Our tools, media, and artifacts are no less situated than we are, and they share the temporality of our existence. They have history, or historicity, or horizons that we merge with our own. The social structure of this shared historicity is something akin to *literacy*, a topic I will return to at length.

Latour's Mediation: Articulations and Translations

Bruno Latour entered the consciousness of English-language social science with his 1979 book with Steve Woolgar, *Laboratory Life*, which has some claim to being the first real ethnography of a scientific laboratory. Latour and Woolgar were interested in explicating the process of scientific investigation, and the tack they took was one which would largely define Latour's career: science as *inscription*—that is, the turning of *things* into *signs*, and, therein, the application of semiotics (in the mode of A. J. Greimas) to science and technology theory.

Latour is now more famous for his association with colleague Michel Callon and the so-called "Paris school" of science and technology studies, and with a school of thought about science and technology called "actor-network theory" (later reified simply as ANT). Actor-network theory has been notoriously controversial (see, for example, Collins & Yearly 1992; Bloor 1999) and, I think, broadly misunderstood—in part simply because the wrong elements are emphasized in the moniker "actor-network theory."

Latour's repeated call has been for symmetry in how we treat human and nonhuman agency, the social and technical, and his attempts at making this clear have required constant re-iteration and

James' Choo-choos

(December, 2003) At 19 months, my son James is enthralled with trains. He has a small wooden train set, and he is fascinated by it, by the way the cars go together, the way they go around the track, the way the track goes together. He looks for trains out in the world and in books. He sees things, like fences, and says, "choo choo," presumably seeing them as if they're tracks.

Trains are educational media for James. Here's how: we cannot assume that a train or trainset is for him what it is for us. A train cannot be said to simply be; a train is the product of heavily layered interpretation, tradition, enculturation. But for James, encountering the world for the first time, a train is something new; he has no idea what a 'real' train is or what it is about—how could he? What a train *is* is so *under-determined* in his case that his understanding of the significance of what a train is must be completely different from, say, mine. We can talk to him about it, because there is a common referent, but its significance in his world is—must be—so very different from mine.

James at 19 months is just beginning to develop an overall sense of the world, as opposed to knowing fragmentary things here and there, the episodes of immediate experience. Now he is beginning to systematize. The trainset—and his railroad trope more generally—is one of the first *systems* of things that he's really engaged with. The train and its microworld is a whole system: it has a grammar, a set of rules, constraints, and possibilities. James uses the train set as a model, or a frame, to look at the rest of the world. The trainset is a language, a symbolic system, and in the way he uses it as lenses with which to see the rest of the world, it is almost entirely metaphoric. Of course, trains are metaphors for adults too, but in a much different, and perhaps less dynamic way.

As he grows up, other things will come to take the place of the trainset as his lens. He learns systems of classifications (animals, colours, weather, etc.) and media like pictures and text. Books and print are already in line to become powerful symbolic tools for his understanding the world, but not yet; a book is still merely a *container* of stories for him, rather than a "personal dynamic medium" like the choo-choo train.

clarification over the past two decades. His first theoretical book in English, *Science in Action,* (1987) made the first broad strokes of a general picture of his ideas. The book was influential but its bold constructivist claims were seen by many as dangerously out on a limb (e.g., Bricmont & Sokal 2001), such that Latour has published substantial reworkings and reframings, notably the excellent essay, "Where are the Missing Masses? The Sociology of Some Mundane Artifacts" (1992), *We Have Never Been Modern* (1993), and *Pandora's Hope: Essays on the Reality of Science Studies* (1999).

There is a general trend in these works from the initial semiotic approach Latour and Woolgar took towards a much more all-encompassing ontological stance, one that bears some resemblance to phenomenological theory. Since the early 1990s, Latour's works have been characterized by his use of a special vocabulary aimed at getting around semantic ambiguities raised by his theorizing, and influenced by A.N. Whitehead's event-oriented *process philosophy*.[2]

In *Pandora's Hope* (1999) Latour provides a comprehensive treatment of technical mediation, presenting the vocabulary of Paris-school science studies—*associations*, *delegation*, *detours*, *goal translation*, *interference*, *intermediaries*, *programs of action*, *shifting in and shifting out*—all of which elaborate Latour's theme of the alliance and alignment of resources, both human and nonhuman, and the ongoing construction of technosocial systems—or networks—thereby.

> Technical artifacts are as far from the status of efficiency as scientific facts are from the noble pedestal of objectivity. Real artifacts are always parts of institutions, trembling in their mixed status as mediators, mobilizing faraway lands and people, ready to become people or things, not knowing if they are composed of one or of many, of a black box counting for one or of a labyrinth concealing multitudes. Boeing 747s do not fly, airlines fly. (Latour 1999, p. 193)

A key motif in Latour's recent writing is that of crossing the boundary *between signs and things*. Two entire chapters of *Pandora's Hope* are devoted to working through this process

2. A.N. Whitehead's 1929 book, *Process and Reality* is the touchstone here; one seemingly picked up by Alan Kay in his early writings as well (see Kay 1972).

in the natural sciences—how, by degrees, raw soil samples become quantified, comparable, publishable inscriptions. And conversely, Latour also attends to the ways in which *signs* are articulated as *things*: he draws attention to the lowly speed bump, which the French call a "sleeping policeman." The speed bump's original admonition, "slow down so as not to endanger pedestrians," becomes by stages translated into, "slow down so as not to damage your car's suspension," and beyond, the message being articulated not in words but in asphalt topography:

> The translation from reckless to disciplined drivers has been effected through yet another detour. Instead of signs and warning, the campus engineers have used concrete and pavement. In this context the notion of detour, of translation, should be modified to absorb, not only ... a shift in the definition of goals and functions, but also *a change in the very matter of expression*. (p. 186)

But, Latour notes, in anticipation of the 'humanist' critique,

> We have not abandoned meaningful human relations and abruptly entered a world of brute material relations—although this might be the impression of drivers, used to dealing with negotiable signs but now confronted by nonnegotiable speed bumps. The shift is not from discourse to matter because, for the engineers, the speed bump is one *meaningful articulation* within a gamut of propositions [which have unique historicity]. Thus *we remain in meaning but no longer in discourse;* yet we do not reside among mere objects. Where are we? (p. 187)

Latour's foregrounding of articulation and translation as key movements in the relationships between us and our material realities makes mediation foundational, and, as with McCullough's practice-oriented framing, it reminds us that mediation is something ongoing, rather than a discrete step or a qualifier of otherwise stable entities. Stability in Latour's writings is an effect, not a starting point; in *Pandora's Hope* he is careful to distinguish between the idea of "intermediaries," which look like objects, and "mediations," which produce them. By working out a vocabulary capable of making such distinctions, Latour

goes much farther than most in giving us a comprehensive philosophical framework for understanding mediation.

Technology as Translation

> *It is in the detours that we recognize a technological act; this has been true since the dawn of time. ... And it is in the number of detours that we recognize a project's complexity.*
>
> – Bruno Latour, *Aramis*

Michel Callon's article "Techno-economic Networks and Irreversibility" (1991) is probably the most lucid single articulation of the position that has come to be called "actor-network theory," which problematizes the individual actor by embedding it in a network and a temporal flow. The network comes to be articulated by means of what Callon calls "displacements"—that is, the re-arrangement and re-definition of various actors' goals, plans, and sub-plans such that these various elements come to be "aligned," forming a network or chain of articulations, and thereby allowing interactions of greater extent, power, and durability. The dynamics of this process are what Callon and Latour spend most of their time explicating; it is a time-consuming business, because each such displacement and alignment depends upon a packaging of complexity into an apparent "black box" of relative stability and dependability. The Paris school locates the methodology of science studies in the unpacking of these boxes, and hence, in the analysis of these networks. And yet, to focus solely on the *nouns* here—actors, resources, intermediaries, networks—is to miss much of the point. Callon's writings also provide a better slogan, one which better focuses on the *process* of articulation and alignment: the *sociology of translation* (Callon 1981; 1986; Callon & Latour 1981).

In taking a complex articulation (it could be the reading of data, the negotiation of funding, the assembly of particular material apparatus or group of people) and rendering it as a *resource* to be used in a larger assembly (with larger/different goals), that particular articulation is *translated* into something more or less different. It has been re-framed and thus re-contextualized, embedded in a different practical or discursive context; in doing so, its

meaning or significance and the work it does changes. This is what is meant by *translation*.[3] Latour explains:

> In addition to its linguistic meaning (relating versions in one language to versions in another one) [translation] has also a geometric meaning (moving from one place to another). Translating interests means at once offering new interpretations of these interests and channelling people in different directions. 'Take your revenge' is made to mean 'write a letter'; 'build a new car' is made to really mean 'study one pore of an electrode'. The results of such rendering are a slow movement from one place to another. The main advantage of such a slow mobilization is that particular issues (like that of the science budget or of the one-pore model) are now *solidly tied* to much larger ones (the survival of the country, the future of cars), so well tied indeed that threatening the former is tantamount to threatening the latter. (Latour 1987, p. 117)

The extent to which these ties are "solid," or, for that matter, "irreversible" is the matter of some discussion within the literature. Callon's 1991 article suggests that successful network alignments (and thus translations) are indeed irreversible, and some of Latour's early writings seem to support this (his use of the term *chreod*—Greek for 'necessary path'—taken from biologist Waddington, confirms this reading. See Latour 1992, p. 240). Later, however, Latour seems to turn from this view, arguing that irreversibility is only the product of continual energy and organization (see Latour 1996 for a full treatment of this theme), that systems or networks may undergo crises at any point which tend to misalign these same resources, turning tidy black boxes (the building blocks of more complex assemblies) back into complexities themselves. Translation is thus more a process than a product; it is "the mechanism by which the social and natural worlds progressively take form" (Callon 1986, p. 224).

The framing of technology as mediation I offered earlier means, to use Latour and Callon's rich keyword, technology as *translation:* the (re-)articulation of the world in new forms and contexts, thereby effecting *transformations* of its 'Being'. Now, if technology is translation, isn't something always "lost in translation"? Of course it is. My appropriation of

3. Latour and Callon credit French philosopher Michel Serres with this usage of "translation."

the term in the service of this exposition is intended to do a particular kind of work; it foregrounds some aspects of a theory of technology and deprecates others. A quote from Callon and Latour shows this in all its political and metaphorical richness.

> By translation we understand all the negotations, intrigues, calculations, acts of persuasion and violence, thanks to which an actor or force takes, or causes to be conferred on itself, authority to speak or act on behalf of another actor or force. (Callon & Latour 1981, p. 279)

Let me leave this particular line suspended for a moment, however, while we return to a mundane example in order to work through some of the ways in which technology as translation can be articulated. Let us take the old shopworn example of the hammer.[4] A translation-oriented way of looking at the hammer is that it is the technology that *translates a nail into a fastener*. This lands us *mise-en-scène* in a network of other articulations: a pointy stick of steel is translated into a nail; a nail becomes a fastener of wood; it translates two pieces of wood into a construction. Now that we have those pieces, the hammer can translate the arm into a nail-driver; the hammer-wielder is translated into a carpenter; a stack of lumber is translated into a house-frame; and so on. A whole concert of temporal displacements and ontological shifts occurs in the rendering of a hammer and a nail into 'functional' pieces. Some of these translations are more permanent than others: the house frame hopefully has some stability; the carpenter's professionalism and income are likely dependent on that translation having some extent in time. The hammer variously translates us into builders, the things it hits into fasteners, and the things we hit nails into into artifacts. Heidegger's point about a different ontology being revealed when we hit our thumbs is true, but what the hammer does when we hit nails is much more important. As Latour takes pains to point out, what is at stake are longer and more durable chains of associations.

Now, as we have all heard, *when all one has is a hammer, everything looks like a nail*, and this is nowhere so true as when musing about technologies. Lest we treat all technology as a

4. Heidegger's famous rendering makes the hammer "ready-to-hand" in the practice of hammering. McLuhan's framing makes a similar move, making the hammer an extension of the arm (or fist, depending which political nail one is trying to strike).

hammer, remember that the hammer and nail example is but one set of articulations; long pants and thatched roofs and texts and numbers are all technologies, too, as are personal computers and nuclear reactors. Each is a particular articulation of networks of varying degree. In each case, different translations are effected. The common thing among these is not the kind or scope of changes which are brought about, but that the fundamental dynamic is one of translations, and chains of translations. Latour goes so far as to deconstruct the divide between so-called 'modern' and 'pre-modern' societies along these lines; what we see, rather than some kind of quantum difference, is a difference in degree, largely expressed in the scope and scale of translations which can be effected, and thus the relative length of the networks that can be sustained (Latour 1987, 1993). Latour offers the example of the French explorer Lapérouse, who visited the east Asian island of Sakhalin briefly in 1787, ascertained from the people living there that it was in fact an island (and not a peninsula extending from Siberia, as the French suspected), and was able to record this knowledge in descriptions sent back to France (Latour 1987, pp. 215–218). There is no qualitative divide between the thinking of Lapérouse and the people of Sakhalin (who, Latour notes, drew maps in the sand for the French), but there is a considerable difference in their relative ability to translate knowledge into "immutable, combinable, mobile" forms (p. 227) which in turn facilitate longer and longer-lasting chains of resources.

Writing, cartography, and celestial navigation made up part of the technology of translation for 18th-century French explorers; so too did muskets, cannons, and square-rigged ships. Latour's account of 'modernity'—though he explicitly forswears the very notion—is one of proliferations of translating technologies, and the longer and longer networks that result. If there is an essence of 'modern' technology, it surely has this characteristic: that it is more deeply intertangled and has more components than seen before. But, Latour insists, this is a difference of degree, not of kind. Translations have been the business of human culture from the Tower of Babel on forward, and the proliferation of them in the modern world is nothing new, in essence.

This is a point upon which Latour departs from many technology theorists who hold that modern technology is oppressive, totalizing, or one-dimensional. The intersection of these ideas yields interesting questions; in particular, Heidegger's classic examination of the *essence* of technology, "The Question Concerning Technology" (1953/1993), could be seen as a translation-oriented approach, with Heidegger's concept of "Enframing" as the master translation, in which human agency is itself translated into a means-and-ends rationality. But Latour rejects Heidegger's conclusions on anti-essentialist grounds (Latour 1993, p. 66), claiming that Heidegger grants far too much power to "pure" instrumental rationality (which Latour characterizes as a myth of modernity), and that actual practice is far more complex than essentialist philosophy admits. Let alone an essence of technology to which we have succumbed, he writes,

> The depth of our ignorance about techniques is unfathomable. We are not even able to count their number, nor can we tell whether they exist as objects or assemblies or as so many sequences of skilled actions. (1999, p. 185)

The alignment of networks and the maintenance of translations is difficult work, Latour argues, but it applies across all facets of culture and society: "...it is no more and no less difficult to interest a group in the fabrication of a vaccine than to interest the wind in the fabrication of bread" (1987, p. 129). In his extended narrative on the aborted development of *Aramis*, an ambitious new rapid-transit system in Paris, Latour waxes eloquent about the difference between the actual and the potential:

> The enormous hundred-year-old technological monsters [of the Paris metro] are not more real than the four-year-old Aramis is unreal: they all need allies, friends, long chains of translators. There's no *inertia*, no *irreversibility*; there's no *autonomy* to keep them alive. Behind these three words from the philosophy of technologies, words inspired by sheer cowardice, there is the ongoing work of coupling and uncoupling engines and cars, the work of local officials and engineers, strikes and customers. (1996, p. 86)

Ongoing work is what sustains technosocial systems over time, what makes them necessarily collectives of both human and non-human actors, and what makes them complex (if not

genuinely chaotic), and thereby eluding essentialist reduction. This is what makes "translation" a better watchword than "actor-network." But remember that translation is fraught with power dynamics; the business of arranging the world into longer chains of mobilized (and therefore transformed) actors exacts a price.

Standardization and the Tower of Babel

In the sociology of translation, the key dynamic in the extension and sustenance of techno-social networks/systems is the translation of heterogenous and complex processes and articulations into seemingly simple "black boxes" which can be effectively treated as single, stable components. Callon wrote that "the process of punctualisation thus converts an entire network into a single point or node in another network" (Callon 1991, p. 153). It remains possible to open these black boxes and reveal the complex details within, but what makes for durable (or, by extension, "irreversible") associations is the extent to which we gloss over their internals and treat a whole sub-process or sub-assembly as a single object. Latour expresses it by making a distinction in his vocabulary between "intermediaries" and "mediations" (Latour 1996, p. 219; 1999, p. 307). *Intermediaries*, which appear as actors, are neat black boxes; *mediations*, which appear as processes, are open, complex, and irreducible to a particular role or function. The two terms are different aspects of the overall process of articulation, viewable as if from above and from below.

The process of making complex details conform to stable black boxes which can then be combined and exchanged is precisely that of *standardization*—a Janus-faced concept[5] that has been the travelling companion of translation since the prototypical technological project, the Tower of Babel. The double loop of translation into black boxes, standardization, and subsequent translations has a particularly interesting implication for technology: it introduces (or reveals) a *semiotic* character to the technical. All technology—on this view— is information technology, or, to put it in the converse, information technology is the paradigm for all consideration of technology.

5. See Bowker & Star's 1999 *Sorting Things Out: Classification and its Consequences*, for a treatment of this dynamic.

This view of technology puts it firmly in a *linguistic* frame, rather than, say, an 'economic' one concerned with commodification, or a 'political' one concerned with domination. It is a view very much influenced by Latour's explication of the "circulating reference" of signs and things. It is (or should be) recognizably McLuhanesque, insofar as it once again puts *mediation* first. It is a view which puts the history of technology in a particular light: the material rendering and transformation of any artifact is inseparable from its symbolic renderings and transformations. Technology is, in this light, *a language of things*. This is but one way to look at technology, but it is one which I think has much to offer to the present study.

Technology is fundamentally and essentially about translation: the symbolic and material rendering of something into something else. Note that the *symbolic* is the primary term here, the *material* is secondary; this makes me an idealist and not a materialist, I suppose, but I mean it this way: technologies are symbolic means of re-ordering the world; in this sense they are just like language.

The Mechanics of Text

> *Lewis Mumford has suggested that the clock preceded the printing press in order of influence on the mechanization of society. But Mumford takes no account of phonetic alphabet as the technology that had made possible the visual and uniform fragmentation of time.*
>
> – Marshall McLuhan, *Understanding Media*

The view of technology I am elaborating here puts the development of the phonetic alphabet as *the* defining technology—at least the defining technology of the Western cultural tradition I can claim to inherit. Language, writing, and standardization of print are all advances in terms of greater and greater *translatability*; the phonetic alphabet is itself the longest-lived and farthest-reaching of all such technologies. Since its original development by the Phoenicians in the third millennium BC, it predates most living languages, has spanned the lifetimes of any number of particular writing systems, and is by far the most influential standardizing principle in Western history. The alphabet underlies all our most

important machines: from books and clocks to computers and networks (McLuhan 1962; 1964).

A theory of technology that locates its essence in translation—that translatability is the *telos* of all technology—is in principle alphabetic, in the various senses that McLuhan outlined: the alphabetic paradigm leads to systematization, organization, ordering schemes. The alphabet is the prototype for standardization: for interchangeable parts and mass production. The alphabet anticipates not just literacy and printing and empire, but mathematics, algebra, mechanization, the industrial revolution, the scientific revolution, and, *par excellence*, the information revolution.

> Only alphabetic cultures have ever mastered lineal sequences as pervasive forms of psychic and social organization. The breaking up of every kind of experience into uniform units in order to produce faster action and change of form (applied knowledge) has been the secret of Western power over man and nature alike. ... Civilization is built on literacy because literacy is a uniform processing of a culture by a visual sense extended in space and time by the alphabet. (McLuhan 1964, pp. 85–86)

What McLuhan realized so very early on was that all technology is information technology; by extension, *digital* technology is the epitome of technology, because digital technology makes the relationship between texts and machines *real* in *real time*. I would like to take a moment to explain what that means.

The equation of texts and machines is more than just a convenient metaphor. I hope to show that it is (has always been) 'literally' true, and that this becomes clearer and clearer with the development of digital technology. I have argued that technologies are essentially about translation—about the symbolic and material rendering of something into something else. This is true of hammers and pillowcases and tea cozies; but it is especially true (or, more accurately, it is especially apparent) of technologies of representation. In late alphabetic culture, we have developed an enormously powerful toolkit of representational technologies: from narrative, we have expanded the repertoire to include accounts, algorithms, arguments, articles, equations, mappings, proofs, tables, theorums, theories,

transactions, and so forth. All of these technologies are representational, obviously, but to put it more forcefully, they are all technologies of translation; they are all machines for capturing and changing the rendition of the world in some measure. We don't think of them as machines because they operate relatively 'quietly'; in our post-industrial imagination, steam power is still our paradigmatic case of mechanization, desite this particular technology's relatively brief lifespan. But perhaps the old trope of the "mechanistic universe" is giving way to one of a textual universe. In the latter part of the 20th century, biologists began to recognize technologies for translation in the mechanisms of cells, and the field of bioinformatics has sprung up around this. Here is another order of information-rendering machines, built of proteins instead of ink or silicon, but recognizable if not yet/quite interpretable.[6]

Language, writing, and standardization (and therefore mechanization) of print are all advances in terms of greater and greater translatability. Numbers, mathematics, and especially algebra are enormously powerful technologies of translation. For example, trigonometry is the translation of the idea of a *circle* into a number of relationships between parts of *triangles* such that both the circle and the triangle can be seen and related in new ways—facilitating the extension of technosocial networks of greater extent (quite literally to the moon and back).

On 'Abstraction'

I want to take a moment here to address the popular notion that information technologies lead to (or are achieved by) greater and greater *abstraction*. Jean Lave (1988, p 40ff; Lave & Wenger 1991, p. 104) and Bruno Latour have gone to lengths to directly warn against this simplistic concept. Rather, the concept of translation renders *abstract* and *concrete* as different languages; it does not set these up in an absolute hierarchy; to do so is to revert to

6. The qualifier here is not intended to soft-pedal a technological determinism, nor to problematize our relative proximity to instrumental power over the genome. Interpretation is *always* not yet/quite possible. Genomics and bioinformatics have shifted this dynamic to new terrain, but I am not at all sure that the core challenge is so different from interpreting written literature. Haraways' extensive (1997) critique of "gene fetishism" makes the same kind of contextualist argument against literalism that literary critics have mounted against determinate formalism. See e.g. Fish 1980.

the old vertical structuralist logic of signifier/signified that I want specifically to avoid. Abstraction is a dangerous metaphor, as Latour notes:

> The concrete work of making abstractions is fully studiable; however, if it becomes some mysterious feature going on in the mind then forget it, no one will ever have access to it. This confusion between the refined product and the concrete refining work is easy to clarify by using the substantive "abstraction" and never the adjective or the adverb. (Latour 1987, p. 241)

Technologies like the alphabet and the computer don't work because of abstraction; if anything, they are effective because of their greater concreteness. It can be argued that the turning point in the development of the digital computer, bringing together the thinking of Alan Turing and George Boole, was when Claude Shannon finally achieved a *sufficiently concrete* means of representing logical relationships. That there exists a pervasive fetishism of abstraction, especially in technoscience, is not to be forgotten, however—to the point where Turkle & Papert (1991) were led to argue for a "revaluation of the concrete." The concrete was of course there all along, despite the mythology of 'abstraction.' The point I want to underscore here is that the dynamics of translation in alphabetic culture are not between brute concreteness and fluid abstractions, but rather between *different forms of concreteness*, lending themselves to different kinds of practices.

Digital translations

Different media are variously translatable; different *genres* are variable in their affordances too. Written text has been the very fount of translatability (in all senses of the word); image less so, and performance less still (hence the oft-quoted—and as oft-misattributed—"talking about art is like dancing about architecture"). The alphabet lends itself to translation by reducing the *material* of representation to a couple of dozen glyphs that can be assembled and reassembled in a famously infinite number of ways. Digital computing extends this reduction by limiting the material of representation to just two states. The result is that 'everything' is renderable digitally (hence, in a crude sense, "multimedia"): a step beyond what could be accomplished with writing, which can only practically render what could be spoken or counted. But the extent to which text, image, and performance can be "digitized" is dependent upon the facility with which we can interpret (that is, translate) these as digital patterns: arithmetic systems were the first to be digitized, back in the 1940s; text-based media came next, and has been so successful as to suggest a revolution in reading, writing, and publishing. Still images are now in widespread digital form; music (both as digital recording and as digital notations like MIDI) has presented few challenges—the digital representation of music is by now the default format. Moving image (video) has been technically difficult (owing largely to the sheer volume of bits that it requires), and the digitization of performative genres like dance has been the least widespread digital form, though certainly not untried.

What is Digital?

"Digital" simply refers to digits, and what are digits but fingers? Digital literally means counting on your fingers, assigning a finger to each thing counted. In the sense of a computer, this is exactly true, except the computer has only one finger, so it counts in 1s and 0s.

A commonly encountered criticism of computers and digital technology is that everything is reduced to an either-or distinction. This is as vacuous as saying that everything in English literature is reduced to 26 letters. But this simplistic critique is based on a misplaced correspondence theory of meaning. If we remember that meaning is in the interpretation, rather than the representation, we quickly get beyond this into more interesting terrain.

But even mechanically, we make up more complex representations than one/zero or yes/no by collecting bits into larger patterns, or in establishing prior context—exactly as we do with words and sentences. As the layerings proliferate, we gain more and more expressive power, and more demands are made on the interpretive system—just like with written literature. And, as we will see in the concept of "late binding," interpretation can be postponed almost indefinitely—just like in post-structuralism!

What is not so obvious here is whether a particular medium lends itself to digital representation as a result of the material affordances of that medium versus our ways of thinking about particular *genres*. For instance, in the past few years, academic journal articles have been largely re-realized in digital rendition and distributed via digital networks, to the point of near ubiquity today. But the relative acceptance of such a shift in this particular genre is in marked contrast to the well-hyped 'e-book' idea, which has promised to move novels and other popular literature into digital formats. That the latter has failed to appear on any significant scale has nothing to do with the material aspects of the medium—novels are composed of the same stuff as journal articles—the difference is cultural: of genres and practices. Genres, of course, do not operate only in the space of literature; they also form the substructure of technological practice. What is interesting is the examination of how both literary genres and technical ones wax, wane, transform, and persist variously in response to the dynamics of media ecology.

But despite the endless (and endlessly interesting) vicissitudes of genre, digital representation implies that everything is translatable into everything else, since the underlying representational mechanisms of all kinds of digital media are common and as simple as possible. The poststructuralist credo that there is nothing outside the text becomes true in a very literal sense: everything becomes a 'text' because everything is, in a literal sense, a text; the genome is but one famous example. The translation of phenomena to textual renderings (which can then be further translated) is the core of what we call science, says Latour; his detailed description of the life sciences show a complex but systematic process of movement along the continuum "between signs and things" (Latour 1999). By this reading, science is a particular form of reading and writing. Indeed, to anyone's eye, science has certainly spawned particular forms of reading and writing; most 'modern' *document* genres, whether they pertain to matters of biochemistry or to journalism, owe their existence to the kinds of translation processes that have been developed in the sciences. Conversely, *documents* themselves are a very interesting kind of technology, and they do a very particular kind of work (see John Seely Brown & Paul Duguid's 1996 "The Social Life of Documents").

Documents operate at a different level than texts *per se*, in that they are technologies that operate on audiences, rather than on individual readers and writers. And yet, in every document is a text doing its own work. The very idea that there can be numerous layerings of text and document (a complex which we commonly call "discourse") underscores the notion that there are large-scale networks of meaning-making at work. The semiotic analogy is doubled and returned: not only do we have tools operating as signs, we have signs that act like tools as the machinery of text meets the semiotics of publication.

Software

> *Thanks to computers we now know that there are only differences of degree between matter and texts.*
>
> – Bruno Latour, *Aramis*

In their book, *The Machine at Work: Technology, Work, and Organization*, Keith Grint and Steve Woolgar make the first explicit equation of texts and machines that I have been able to find in the sociological literature (though the idea has older roots in literary criticism; see Landow; Aarseth; etc.). What falls out of this equation is that the analogy of reading and writing becomes possible with reference to machines, rendering machines "hermeutically indeterminate" (Grint & Woolgar 1997, p. 70). This is all very well, and no doubt lends some valuable light to the study of various technologies. But Grint and Woolgar decline to take the next step: they persist in talking (as do most technology theorists) of machines in the "steam-engine" sense: as physical mechanisms, made out of hard stuff and powered by shovelling coal or at least plugging in the power cord. Although computing technology figures in Grint and Woolgar's analysis, their attention sticks with the plastic-and-metal object on the desk. It does not venture inside, to *software*.

Even while making a case for the absolute blurriness of the line between texts and actions, Latour too holds tight to the conventional divide between steel and words:

> We knew perfectly well that a black box is never really obscure but that it is always covered over with signs. We knew that the engineers had to organize their tasks and learn to manage the division of their labour by means of millions

> of dossiers, contracts, and plans, so that things wouldn't all be done in a slap-dash manner. Nothing has a bigger appetite for paper than a technology of steel and motor oil. ... Every machine is scarified, as it were, by a library of traces and schemas. (Latour 1996, p. 222)

If there's anything that's been shown in a half century of computing, it is that machines are not reliant on steam and steel. *Machines are pattern-processors.* That one particular pattern is in steel and another in fabric and another in bits is inessential. Where Latour and Grint & Woolgar neglect to go is precisely where I *do* want to go: the machine is text—and this is not just an analogy that makes literary techniques applicable. Machines *are* literature, and software makes this clear. This may not be yet/quite apparent in the public imagination, owing largely to our collective hardware, or *gadget*, fetishism. But the argument for placing the focus of our technological inquiries at the software level rather than at the hardware level is very strong. Pioneering computer scientist Edsger Dykstra wrote, in 1989:

> What is a program? Several answers are possible. We can view the program as what turns the general-purpose computer into a special-purpose symbol manipulator, and it does so without the need to change a single wire... I prefer to describe it the other way round. The program is an *abstract* symbol manipulator which can be turned into a *concrete* one by supplying a computer to it. (Dijkstra 1989, p. 1401 [italics added])

The efficacy of this perspective has been apparent within computer science since the late 1950s, and in particular, since the advent of John McCarthy's computer language *Lisp* (McCarthy 1960), hailed as doing "for programming something like what Euclid did for geometry" (Graham 2001).[7] The significance of Lisp and the ways of thinking Lisp ushered in have been obscured by the popular rendering of Lisp as "an AI language" and therefore subsumed within the quest for artificial intelligence. But Lisp's connection with AI is an "accident of history" (Graham 1993), one which I will not dwell on here. What I do want to foreground here is the idea of textual constructions—programs and programming languages—acting as machines in their own right. Of course it is possible to quibble with

7. Alan Kay called McCarthy's contribution the "Maxwell's equations of software" (Kay & Feldman 2004)

Dijkstra's formulation and strike a hard materialist stance,[8] insisting that the electronic circuitry is the machine and that programs are "superstructure." What McCarthy's Lisp provides is a solid example in which it clearly *makes more sense* to view it the other way around: that the real machine is the symbolic machinery of the language and the texts composed in that language, and the details of the underlying hardware are just that: details. A half-century of Lisp[9] provides considerable evidence that the details steadily decrease in importance. In 1958, when the first implementation was made, it was of course a matter of enormous resourcefulness and innovation on the part of the MIT Artificial Intelligence lab, and so it remained, in rarified academic circles well into the 1970s, when implementations began to profilerate on various hardware and as Lisp became core computer science curriculum at MIT and other institutions (see Steele & Gabriel 1993; Abelson & Sussman 1996). Today, downloading and installing a Lisp implementation for a personal computer is a matter of a few moment's work.

But more importantly, as Paul Graham's (2001) paper "The Roots of Lisp" demonstrates for modern readers (as it is a reworking of McCarthy's original exegesis), Lisp's core simplicity means it doesn't require a 'machine' at all. Graham's paper (it is important to dwell for a moment on the word "paper" here) explains how, in the definition of a dozen or so simple functions—about a page of code—it is possible to create a system which is a formal, functional implementation of itself. The machinery of Lisp works as well in the act of reading as it does in digital circuitry.

8. Friedrich Kittler famously made this move in his (1995) essay, "There is No Software,"—ignorant of both Dijkstra and McCarthy, as far as I can tell—in which he argues that everything is indeed reducible to voltage changes in the circuitry. Kittler's argument is clever, but I don't find that it actually sheds any light on anything. It is rather *reductio ad absurdum*, leaving us with no better grasp of the significance of software—or any means of critically engaging with it—than a study of letterforms offers to the study of English literature.
9. That Lisp has been around for half a century provides us with a near-unique perspective on the evolution of computing cultures; hence my argument for cultural history of computing.

The significance of digital computing—that which systems such as McCarthy's Lisp make so clear—is *not* how computers have been brought to bear on various complex *information processing* applications (calculating taxes, computing ballistics trajectories, etc.). Nor am I about to claim that the digital revolution brought forth AI and electronic autopoeisis, nor will I allude to any other such Frankensteinian/Promethean narrative. The far greater significance of digital computing is in the use of alphabetic language as a kind of "bifurcation point" (to borrow the language of systems theory), at which a different level of order emerges from the existing substrate. The advent of digital computation marks the point at which the text/machine equation becomes literally and manifestly real. Invoking systems theory in this way implies that the elements of this shift are largely internal to the system; it was not the advent of the semiconductor, or the pocket protector, or any such isolable, external factor that led to this shift. Rather it has more to do with the ongoing configuration, reconfiguration, and 'refactoring' of alphabetic language.

On Lisp

The simplicity of Lisp has to do with its straightforward alphabetic nature. Lisp was, and is, a formalism expressed in written language. It is mathematical, to be sure, and its creators were mathematicians, but Lisp was about symbol manipulation rather than arithmetic calculation. McCarthy "used mathematical formalisms as languages and not as calculi" (Stoyan 1984).

Paul Graham, a present-day Lisp evangelist, writes, "It's not something McCarthy designed so much as something he discovered. It's not intrinsically a language for AI or for rapid prototyping, or any other task at that level. It's what you get (or one thing you get) when you try to axiomize computation." (Graham 2001, p. 11).

Lisp is hailed as the second-oldest programming language still in use today; the oldest is FORTRAN, IBM's language for numerical calculation, still used in scientific computing applications.

The structure and syntax of Lisp programs is deceptively simple: everything is a list (Lisp is short for "List Processing"), enclosed within parentheses. A core function called 'car' provides a mechanism for splitting the first item from a list; recursively applying such a function allows one to traverse lists (and lists of lists, running to indefinite depths of nesting). Other core functions provide means of comparing the evaluations of items, and for making conditional statements. And then, almost literally, everything else is created from these primitive building blocks, by layering and layering larger and larger evaluation structures.

McCarthy's original innovation is described in a conference paper (1960) of thirty-odd pages. In it, there are no circuit diagrams or instructions for constructing electronic devices. Rather, it is a concise work of symbolic logic, describing the elements of a formal notation for describing recursive systems which are, interestingly, capable of describing themselves. McCarthy's work on the language in the late 1950s largely preceded any working implemen-

tation: "I decided to write a paper describing Lisp both as a programming language and as a formalism for doing recursive function theory" (1981). McCarthy was after a realization of Alan Turing's formal theory of computability, and Lisp was a success in this regard; it was a practical implementation of the use of recursive functions as an equivalent to the Turing machine (which, although logically complete, is not a practical system—see Graham 2001, *n*5). Lisp's legacy is thus not in electronics so much as in logic and formal systems—in *language*.

The Semiotics of Standardization

It would be trite to say that this innovation had been in the works for some time. I have mentioned Alan Turing, but other significant contributions—such as those of logician George Boole and the famous team of Charles Babbage and Augusta Ada—were required in order for the story to unfold. These three are key to this telling of the story precisely because their contributions predate any workable physical instantiation of the machine/text that their names would come to symbolize.

Babbage is particularly interesting in this respect, precisely because his works were not practically realized despite his best efforts. One of the most intriguing subplots in the history of computing is that it was indeed *practically* impossible, in Babbage's day, to create a calculating machine of the complexity he required, owing to the relative lack of manufacturing sophistication. Specifically, it was not yet possible, in the mid 19th century, to manufacture the thousands of gears required for Babbage's "Difference" and "Analytical" engines with sufficient precision—that is, to sufficiently close tolerances—that such a machine could actually run. What was required was the advent of standardization of gear cutting and manufacturing apparatus, a fact not lost on Babbage, who published significant works in this area. Interestingly, the standardization of gear cutting (and also screw threads) was largely pioneered by Joseph Whitworth, an engineer who had worked with Babbage, but the required precision was ultimately achieved in the 20th century, not the 19th. It has thus become possible—even relatively economical—to create working instances of Babbage's

machines (indeed the Science Museum in London, England has done so), given modern-day manufacturing (Doyle 1995).

In considering this side-story, I want to draw the focus again to the symbolic rather than the material. The difference between gear cutting in the 19th century and in the 20th isn't merely that the tolerances are finer; rather, the key is standardization, mass production, and the translation of the artifact from *individual incarnation* to its status as a *commodity*. What happens when you standardize production is that you shift the artifact semiotically. It stops being a isolated signifier of its own, and starts being a neutral component—that is, a "black box"—that can be assembled into larger systems, just like the letters of the alphabet. The glyph itself, like the gear, stops being a thing-in-itself and starts being part of a larger semiotic apparatus. This, according to Havelock (1980), is precisely what happened when the Greeks began to distinguish between consonants and vowels in written language, thereby making a more analytically complete mapping of morphemes to glyphs. The result was that the *individual letters became unimportant* in comparison to the words, sentences, and paragraphs that were built out of them—this is evident in constrast with, for instance, the letter-oriented focus of the Kabbalistic tradition, in which significant meaning is vested with the individual letters themselves (Drucker 1995, p. 129). This is also, arguably, paralleled in the evolutionary shift from single-celled to multi-celled organisms, in which the activities and identitiy of individual cells becomes subsumed in the structure of the larger organism. Standardization of alphabets, of currencies, of machinery, even of living structures, all effect this kind of shift towards a (analytically) higher-level assemblages of order and agency.

There is clearly a moral question which emerges here, since we are not merely speaking of dumb objects, but conscious subjects too. Bowker & Star problematize standardization thusly:

> We know from a long and gory history of attempts to standardize information systems that standards do not remain standard for very long, and that one person's standards is another's confusion and mess [. . .] We need a richer

> vocabulary than that of standardization or formalization with which to characterize the heterogeneity and the procedural nature of information ecologies. (Bowker & Star 1999, p. 293)

Bowker & Star's plea, however, comes at the end of their excellent book on "categorization and its consequences" and not at the beginning, and so we are left with no more than an opening made toward the study of process, articulation, negotiation, and translation. These dymanics are invoked in a clearly political mode, for these are the dynamics of standardization and classification themselves. Compare Donna Haraway's description of the digitization of the genome and the subsequent emergence of the field of bioinformatics:

> Yet, something peculiar happened to the stable, family-loving, Mendelian gene when it passed into a database, where it has more in common with LANDSAT photographs, Geographical Information Systems, international seed banks, and the World Bank than with T.H. Morgan's fruitflies at Columbia University in the 1910s or UNESCO's populations of the 1950s. Banking and mapping seems to be the name of the genetic game at an accelerating pace since the 1970s, in the corporatization of biology to make it fit for the New World Order, Inc. (Haraway 1997, p. 244)

What is at issue, it seems to me, is not whether standardization is good or bad (or any similarly framed substantivist argument), but, as Haraway points out: *What counts? For whom? At what cost?*

Simulation as Interpretation

> *"To know the world, one must contruct it."*
>
> – Cesare Pavese, quoted by Alan Kay

The preceding discussion of technology and translation, machine and text, is intended to create a frame for a topic which I will introduce here, but which I want to revisit a number of times in the pages that follow: *simulation*.

Simulation is a paradigmatic application of information technology, something often hidden by our tendency to instrumental reason, but which the sociology of translation helps

to illuminate. I mean this in the sense that simulation can be used as a general motif for viewing and understanding a wide variety of computing applications, from mundane 'productivity applications' to the more stereotypical systems simulations (weather, fluid dynamics, etc.). Alan Kay and Adele Goldberg put it generally and eloquently:

> Every message is, in one sense or another, a simulation of some idea. It may be representational or abstract, isolated or in context, static or dynamic. The essence of a medium is very much dependent on the way messages are embedded, changed, and viewed. Although digital computers were originally designed to do arithmetic computation, the ability to simulate the details of any descriptive model means that the computer, viewed as a medium itself, can be all other media if the embedding and viewing methods are sufficiently well provided. (Kay & Goldberg 1976)

What's interesting and compelling about computing is not the extent to which models, simulations, representations are true, real, accurate, etc., but the extent to which they *fit*—this is the lesson of Weizenbaum's infamous ELIZA, the uncomfortable thesis of Baudrillard's *Precession of Simulacra*, and the conclusion of a growing body of literature on virtual reality. It is also, to step back a bit, one of the central dynamics of narrative, especially in the novel and myriad other literary forms. "Simulation is the hermeneutic Other of narratives; the alternative mode of discourse," writes Espen Aarseth (2004). If *effect* is the important point, then this is by definition an anti-formalist argument. But simulation is not merely reducible to surface and appearances at the expense of the deeper 'reality'—it reflects rather the deeper aspect of Baudrillard's "simulacra"—precisely where we live in today's world. But where Baudrillard's take is bitter and ironic, I have always been fond of Paul Ricoeur's hermeneutic version:

> Ultimately, what I appropriate is a proposed world. The latter is not *behind* the text, as a hidden intention would be, but *in front of* it, as that which the work unfolds, discovers, reveals. Henceforth, to understand is *to understand oneself in front of the text*. (Ricoeur 1991*b*, p. 88)

Madeleine Grumet similarly works with Ricoeur's framing in her discussion of theatre, the "enactment of possible worlds;"

> ...performed in a middle space that is owned by neither author nor reader. Constructed from their experience and dreams, this liminal space cannot be reduced to the specifications of either the author's or the reader's world. [...] Performance simultaneously confirms and undermines the text. [...] Mimesis tumbles into transformation, and meaning, taken from the text, rescued from the underworld of negotiation, becomes the very ground of action. (Grumet 1988, p. 149)

Rather than some sinister command-and-control reductivism, this is the space of simulation. If we go further, and look at simulation—model building—as a hallmark of science and the modern world, Ricoeur's stance presents itself as a refreshing alternative to the two horns of objectivism and relativism (Bernstein 1983). We needn't get ourselves tied in knots about our access to 'reality,' since our business—in science, technology, literature, politics—is fundamentally about *invention* and not discovery, and we can avoid the spectre of relativism, because it is *the world* which we are building, not just arbitrary constructions.[10] "The point is to cast our lot for some ways of life and not others," Haraway admonishes. Thus, it matters greatly *which* constructions we choose; the process of creating them and deciding upon whether or not to embrace them is fundamentally political (Latour 1999). It is not so much *dominated by* issues of power but the very crucible *wherein* power is exercised and contested.

Simulation—that is, model building—is essentially *hermeneutic*. Its process is that of the hermeneutic circle, the merging of the horizons of modeller and world, the part and the whole, and is determined by R.G. Collingwood's "logic of question and answer" (Gadamer 1975/1999, p. 362ff). The model—which is constructed, and therefore concrete—poses the questions to which the 'world'—ultimately inaccessible and therefore uncomfortably

10. Latour's "The Promises of Constructivism" (2003) makes a similarly positive argument.

'abstract'—is the answer. It is not, thus, analytic or reductive, nor is it definitive; rather it is, ideally at least, dialogic.

Writing of the dynamic between simulation and narrative in games, Espen Aarseth says,

> If you want to understand a phenomenon, it is not enough to be a good storyteller, you need to understand how the parts work together, and the best way to do that is to build a simulation. Through the hermeneutic circle of simulation/construction, testing, modification, more testing, and so forth, the model is moved closer to the simulated phenomenon. (Aarseth 2004)

This is decidedly not to say that the model is ever actually *complete*, but that our "foreconception of completeness" (Gadamer 1975/1999, p. 370) is a necessary precondition of participation and engagement. Once again, I want to avoid the vertical correspondence logic of classical structuralism (signifier/signified; model/reality) and instead pursue a vision wherein the elaboration of the model is its own end; it succeeds or fails not by being more or less faithful to 'reality', but by being *better connected* (to invoke Latour and Callon's network model once again). There is undoubtedly enormous danger lurking in the seduction of the model, and we undoubtedly forget again and again that the map is not the terrain, becoming literalists once again. That this is a danger does not imply that we should avoid making models, though, just that we must strive to avoid taking our fetishes and "factishes" (Latour 1999) too literally. Writing on the mapping of the genome, Haraway notes:

> Geographical maps can, but need not, be fetishes in the sense of appearing to be nontropic, metaphor-free representations, more or less accurate, of previously existing, "real" properties of a world that are waiting patiently to be plotted. Instead, maps are models of worlds crafted through and for specific practices of intervening and particular ways of life. ... Fetishized maps appear to be about things-in-themselves; nonfetishized maps index cartographies of struggle or, more broadly, cartographies of noninnocent practice, where everything does not have to be a struggle. (Haraway 1997, pp. 136–137)

It is important to remember that Haraway's argument is not against cartography—it is against the temptation to think that we and our constructions are somehow pure or innocent. There is no shortage of literature warning of the dangers of simulation; Kay himself wrote that "as with language, the computer user has a strong motivation to emphasize the similarity between simulation and experience and to ignore the great distances that symbols impose between models and the real world" (Kay 1977, p. 135). But to contrast simulation with something like "local knowledge," as Bowers (2000) does, is to badly miss the point and to essentialize (and caricature) both simulation and local knowledge. Local knowledge is mediated knowledge too. "Situated" knowledge is nothing if not mediated. We will return to simulation, at length, later.

The Ethics of Translation

> *We are responsible for boundaries; we are they.*
>
> – Haraway, *Cyborgs, Simians, and Women*

> *The new media and technologies by which we amplify and extend ourselves constitute huge collective surgery carried out on the social body with complete disregard for antiseptics.*
>
> – McLuhan, *Understanding Media*

The foundation of my argument is that human culture is fundamentally and essentially technological—that is, technologically mediated. It makes no sense to attempt to isolate what is 'human' from what is 'technical'. As Latour has taken great pains to point out (esp. 1993), the attempt to isolate and purify these—that is, to come up with a society sans technology or a technology sans society—has been at best fruitless. And yet it is a powerful temptation, as we have inherited a weighty intellectual tradition devoted to just such a process of purification. The words we use readily betray it: culture, society, technique. I begin to think that it rarely makes sense to talk of culture—or society—at all; Latour's refocus on the "collective" of humans and nonhumans is really the only sane construction.

Back to the Tower of Babel

Given the core notion of technology as *translation*—as *delegation*, as in one of Latour's rich metaphorical turns—the means of translating or transforming the world is, in a trivial sense, about power. But if the fundamental transformation is on the symbolic level rather than the physical, then this is even more important, for we are speaking of the power to shape people's reality, and not just their physical landscape. Technology is thus the medium of power/knowledge *par excellence*, for it simultaneously establishes and enforces power/knowledge structures (by very definition) and also provides the means for their subversion. This is politics. Haraway, again, has said it most eloquently:

> In short, technoscience is about worldly, materialized, signifiying, and significant power. That power is more, less, and other than reduction, commidification, resourcing, determinism, or any other of the scolding worlds that much critical theory would force on the practitioners of science studies, including cyborg anthropologists. (Haraway 1997, p. 51)

Again, the admonition is to stop worrying about purification of essences, about what the world would be like without the polluting effects of technology, pining for an unmediated reality in some nostalgic reminiscence of a simpler age. Asymmetry in theory and practice is the order of the day—asymmetry of ways of seeing and drawing the world, asymmetry of access to ways and means, asymmetry of expression. But this isn't to be avoided or solved or redeemed in Gordian-knot fashion. Rather, it is to be confronted. Something is "lost in translation" because translation is always interpretation. The only remedy is further interpretation, the ongoingness of the dialogue. So it is with all things political.

Our responsibility to technology

> *Located in the belly of the monster, I find the discourses of natural harmony, the nonalien, and purity unsalvageable for understanding our genealogy in the New World Order, Inc. Like it or not, I was born kin to Pu[239] and to transgenic, transspecific, and transported creatures of all kinds; that is the family for which and to whom my people are accountable.*
>
> – Donna Haraway, *Modest_Witness*

When Everything Looks Like a Nail

The danger of translations can be simply recognized in the old chestnut: When all you have is a hammer, everything looks like a nail. McLuhan's saying, "we shape our tools and thereafter, our tools shape us" is a more formal articulation of this basic point, which is interestingly absent from Heidegger, even though it is a commonplace for us. Heidegger seems genuinely worried that everything in the world (including human beings) has begun to look like nails, but he downplays—to the peril of his argument—the hammer's *generative* role in this.

Larry Wall, developer of the immensely popular open-source programming language Perl (called the "duct tape of the Internet"), made the following insightful comment:

> *You've all heard the saying: If all you have is a hammer, everything starts to look like a nail. That's actually a Modernistic saying. The postmodern version is: If all you have is duct tape, everything starts to look like a duct. Right. When's the last time you used duct tape on a duct? (Wall 1999)*

The challenge, with respect to our fundamental relationship to technology and technological change, and the asymmetries which result, is one of *literacies*. By "literacy" I do not trivially mean the ability to read and write, but rather the ability to enter into and participate actively in ongoing discourse. If, as I have argued, technology is about the symbolic realm at least as much as the physical, then the importance of literacy is not confined to the written word. It is the matter of participating in the discourses which surround us; discourses of power. Technologies—things—are as much a part of discourse as are words. The task before us, in the early decades of a digitally mediated society, is to sort out the significance of a digitally mediated discourse, and what the implications are for actual practice.

Consider the stakes. Without *print literacy*—in the sense in which we readily acknowledge it today as being fundamental to democracy and empowerment and social change—print technology would be nothing but an instrument of oppression. The written word itself, without the concept of a widespread (if not mass) print literacy, represents a terribly asymetrical technology of domination over the world. But it is this literacy, in the broad, distributed, bottom-up sense of the word, that saves the written word from being a truly oppressive development for humanity. Further, that print literacy is instead taken as the

instrument—*and the symbol*—of liberation speaks to a dynamic not accessible from an examination of the material or cognitive features of reading; rather, literacy as an agent of emancipation—which then translates the written word into an agent of emancipation—operates on the larger socio-cultural level. But at this point in the Western world, "technological literacies" are nowhere near as widely spread in practice as their print-oriented conterparts, nor are they held up symbolically as agents of emancipation and democratic participation.

Fullan and Hargreaves (1996), writing of school reform, say, "Teaching is not just a technical business. It is a moral one too" (p. 18). *Technology is not just a technical business either*, and the longer we collectively pretend that it is, as we are wont to do as long as we remain inside our instrumentalist frame, the less we are able to grapple with it, politically and morally.

But as instrumental logic blinds us to the political and moral implications of our mediated practices, so too the rhetoric of technological determinism provides a crippling apparatus with which to reach a critical awareness of technology. In the shadow of our yearning for a pure, unmediated past is the parallel idea that technological mediation is leading us down a tragic path 'no longer' of our own choosing. Of late, the tendency is to see digital media as the handmaiden of globalization (e.g., Menzies 1999). Here the argument is that the rendering (commodification) of all human activity into an easily translatable digital currency is at the expense of "local" culture, of the less privileged, and in the interests only of the corporate sector. It is an easy argument to make, and the argument borrows much from the connection between literacy and colonialism (e.g. Willinsky 1998). However, where the latter argument succeeds is in a far more sophisticated appreciation of the detail: literacy is certainly an agent of homogenization and indeed domination, but it is also, in countless cases, an agent of differentiation and emancipation (e.g., New London Group 1996). Note that this is not the same thing as *resistance* in the sense of resisting the onslaught of print media or computerization. "Resistance" is a troublesome term, because it suggests an "either-or," a one-dimensional problem (as in G.W. Bush's "You are either with us, or you

are against us"). Re-shaping, re-direction, and re-figuration, as in Haraway's rich repertoire, are more apt.

I suggest that print or alphabetic literacy is not inherently an instrument of domination *precisely because the alphabet is open*; in contrast, where state power has controlled who has access to reading and writing, it is not open, and for this very reason, modern democracies enshrine the institutions of mass literacy: public education, freedom of the press, and so on—lest you think my argument is for the perfection of these institutions, I mean here to point to these as shared ideals. But even the curious logic of liberal capitalism seems to realize (on odd days, at least) that in order to be effective, languages and communications must be open—private languages cannot thrive. And in openness is the possibility of refiguration. A student of mine, Bob Mercer, made this point about the unreflexive conceit of the "end of history:"

> If consumption in an industrial society is of industrial goods—cars, refrigerators, televisions, computers—what then is consumed in an information society? Information, surely, and some of that information takes the form of ideas. And some of those ideas in turn challenge the consumer society. (Bob Mercer, 2003, "Blogging at the End of History")

Technology, like language, is not the *instrument* of power; it is the *crucible* of power, the very setting where the contest for one way of life or another takes place. Feenberg's framing of technology as a site of struggle means that the debate is ongoing; it is not an argument to be won or lost, but to be continually engaged. In putting the emphasis on *technology as translation*—of representations, of agency, of apparent worlds—I hope to open up this theoretical arena to what follows, which is an examination of a very particular and very ambitious project to develop a technological infrastructure capable of engagement with high-level philosophical, educational, and political themes.

Chapter 4: Alan Kay's Educational Vision

Alan Kay's project rests upon a number of substantial philosophical and theoretical foundations. An examination of these will be helpful in the analysis of the trajectory of the Dynabook project over the past three decades. The following treatment draws less from the 'primary' documents of the early 1970s as from Kay's own reflection and exegesis—especially that of recent years, which is substantial and which reveals a definite historical self-awareness in Kay's work.

Kay's own sense of his place in history is a theme which emerges repeatedly in his writings, from early grand ambitions of "paradigm shifts" to his more studied reflections on a fledgling digital age in comparison with the advent of print in Europe four or five centuries before. Throughout, Kay establishes his subject position in a decidely Romantic mode—to briefly invoke Hayden White's schema (1973) of historical emplotment—but in the first person and of the first order.

I would like here to present an overview of each of the major themes elaborated in Kay's writings and talks. These are, in brief:

1. The vision of *computers for children*, and the early and foundational influence of Seymour Papert's innovative research with the Logo programming language;
2. *Systems design* philosophy, drawing on insights borrowed from cell biology and American political history;
3. The *Smalltalk* language and the *object-oriented* paradigm in computer science, Kay's most important and lasting technical contribution;
4. The notion that *Doing with Images makes Symbols*, a phrase which embodies an application of the developmental psychology of Jerome Bruner;
5. *Narrative, argumentation, and systems thinking*; different modalities for expressing truths about the world;

6. A particular *conception of literacy* that broadly includes technological mediation as a cultural and historical force.

Computers, Children, and Powerful Ideas: The Foundational Influence of Papert

The image of children's meaningful interaction with computers in the first place evokes the image of MIT mathematician and computer scientist Seymour Papert, his work with the *Logo* programming language for children, and his influential writings on the role of computing in education. Papert's research began in the late 1960s with Wally Fuerzig, Danny Bobrow, and Cynthia Solomon at the Massachussetts Institute of Technology and private contractor BBN[1] (Chakraborty et al. 1999). Papert was one of the founders of the Artificial Intelligence Lab at MIT, but had previously studied children's epistemology with Jean Piaget in Geneva—though his pedagogy drew heavily on John Dewey and Maria Montessori as well. Papert's idea that computers could be used to help children gain an embodied or concrete understanding of what is more commonly taken as a symbolic, formal mode of thought—mathematics—draws directly from Piaget's developmental psychology. This basic idea was a major current—perhaps *the* major current—in educational computing by the early 1980s, and Papert's Logo programming language embodied this particular philosophy and method. Papert and Alan Kay had an early influence on one another, and their ideas have been entwined for over three decades, though Papert is clearly the more famous of the pair. Since the Papert side of the story has been interpreted and reinterpreted by generations of scholars already,[2] I will approach the story from Kay's perspective.

Alan Kay visited to see Seymour Papert and his team in 1968; the meeting was reportedly a life-changing one for him. Papert, Cynthia Solomon, and Wally Feurzig had begun

1. Cambridge, MA-based BBN (Bolt, Beranek, & Newman) is wrapped up intimately in the history of computing. Most famously, the company managed the implementation of the original ARPAnet in 1969.
2. Papert's own writings (1980a; 1972/1980b; 1987; 1991; 1992) are the core texts of the Logo philosophy, a literature fleshed out extensively by his colleagues and graduate students (e.g., Turkle 1984; Solomon 1986; Harel & Papert 1991; and many others). A mid-1980s "Logo backlash" appeared, summarized in the edited collections by Sloan (1985) and Pea & Sheingold (1987), though substantially re-addressed in Noss & Hoyles 1996, as well as Papert's own later writings. Latter day reflection and criticism can be found in Chakraborty et al. 1999; diSessa 2000; and Aglianos 2001.

working with children in schools in Lexington, MA with the Logo language. At the time, Logo was used to control a robotic "turtle"—a half-metre plexiglass dome on three wheels that drew lines as it crawled around big pieces of butcher paper. Papert's concern was with "teaching children thinking" (1972/1980b)—that is, "really thinking about what they do" (p. 161). Papert's project was to make mathematics *real* for children, instead of a difficult formalism. In 1972, he wrote:

> Mathematics is the most extreme example. Most children never see the point of the formal use of language. They certainly never had the experience of making their own formalism adapted to a particular task. Yet anyone who works with a computer does this all the time. (p. 162)

The process of working systematically with Logo to make the turtle draw geometric patterns, said Papert, had the effect of getting children to "think mathematically." They were thus able to get *inside*—to physically embody, as it were—the mathematical and geometric constructs.[3] Alan Kay was impressed: "Here were children doing real programming," he remembered, and this "finally hit me with the destiny of *what personal computing really was going to be*" (Kay 1996*a*, p. 523 [italics added]). The insight that struck Kay was that children would be the critical users of personal computers when they ultimately became available; therefore *children* were who they should be designing for.

3. Papert specifically wrote of "the development of an ego-syntonic mathematics, indeed, of a 'body-syntonic' mathematics" (1980*a*, p. 205). "Like the child, it is always at the center of its universe," wrote Kay (1990, p. 194) of the Logo turtle's local coordinate system (as opposed to a Cartesian one).

In his 2002 lecture, *The Computer Revolution Hasn't Happened Yet*, Kay noted:

> Papert had kids doing math—real math that you use in science, for ten-year-olds, via playing, Montessori-style. . . . This was just the best idea anybody had ever had for computing.
>
> Papert's thing had real zen behind it. Not just math and science, but the zen of math and science. . . . You don't automatically get zen by learning how to program in Logo, but Papert correctly saw that one of the greatest vehicles anybody ever came up with for getting enlightened was the computer. (Kay 2002*a*)

Kay's work in the late 1960s had been dominated by his graduate research project on *The Reactive Engine* (Kay 1968), an embryonic personal computer called the "FLEX Machine," which attempted to generalize the *master and instance* architecture of Ivan Sutherland's computer graphics research from the early 1960s,[4] and to elaborate a vision of what might be the successor to the dominant computing paradigm of the day: time-sharing terminals connected to a central, large mainframe computer.[5]

Some Logo Examples

Here is some Logo code for drawing stars with the turtle, and the trail of a discovery:

```
repeat 5 [forward 70 right 150]
```

```
repeat 5 [forward 70 right 140]
```

```
repeat 5 [forward 70 right 144]
```

```
repeat 5 [forward 70 right 720/5]
```

```
to fivePointStar
  repeat 5 [
    forward 70
    right 720/5
  ]
end fivePointStar
```

4. Ivan Sutherland was Kay's graduate supervisor at the University of Utah in the 1960s. Sutherland was famous for his work on SKETCHPAD, the original interactive graphics application (Sutherland 1963). A major feature of Sketchpad was its capability to treat a complex diagram as a "master," from which "instances" or copies could be instantly created, each one bearing all the features of the original; a change made to the master drawing would be immediately reflected in all instances of it; this is the prototype for the "class-and-instance" architectural model of most object-oriented systems from Kay's Smalltalk forward.

But the FLEX Machine's intended users were adult professionals, and Kay later called its user interface "repellent" (Kay 1987).

When Kay was recruited to Xerox PARC by ex-ARPA director Robert Taylor, his first project was the design of a computer he called *KiddiKomp*, a prototype desktop-based machine based around a 12" Sony Triniton screen. With his newfound sense of the "real audience" for personal computing, KiddiKomp drew not only on Kay's work on the FLEX Machine, but also on Papert's Logo. Logo's influence on Kay's software designs in the early 1970s is considerable; there have been turtle graphics in nearly every Kay-designed software in the past 30 years (Goldberg & Kay 1976; Gillespie 2002). But Kay's version of computers for kids goes beyond turtle geometry (as does Papert's, despite popular conceptions). In 1972, Kay spoke of the breadth of his vision:

> This new medium will not "save the world" from disaster. Just as with the book, it brings a new set of horizons and a new set of problems. The book did, however, allow centuries of human knowledge to be encapsulated and transmitted to everybody; perhaps an active

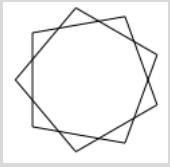
Generalizing by adding a variable *n*:

```
to star n
  repeat n [
    fd 100
    rt 720/n
  ]
end star
```

```
star 5
```

```
star 7
```

```
star 9
```



```
star 8
```

Hmm... why not an 8-sided star?

5. Time-sharing systems formed much of the context for computing research in the late 1960s,especially within the ARPA project. This notion of several interactive terminals (teletype units;later keyboards and screens) connected to a single, powerful mainframe computer was an enormous shift from the batch-processing model which preceded it; in time-sharing, the computer shared its users simultaneously rather than one-at-a-time as in batch processing. Time-sharing systems (and their recognizable terminals) are still very much with us today, especially in point-of-sale systems.

> medium can also convey some of the excitement of thought and creation! (1972, p. 1)
>
> Where some people measure progress in answers-right/test or tests-passed/year, we are more interested in "Sistine-Chapel-Ceilings/Lifetime." (p. 4)

This 1972 paper, entitled "A Personal Computer for Children of All Ages" is Kay's original manifesto for computers in education. It is the first published work outlining his newly oriented project and the ideas underlying it. It begins with a discussion of schools and school reform, but switches quickly to a scenario involving two young kids, Jimmy and Beth, who embody the kind of exploratory, constructivist learning he saw as ideal for the coming century. Jimmy and Beth, working with their "Dynabooks"—lightweight portable computing devices—stumble upon a question about gravity while playing a video game.

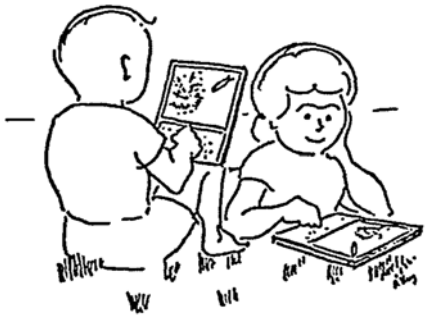


Figure 4.1: Jimmy and Beth with their Dynabooks. (from Kay 1972)

Jimmy and Beth proceed to consult their teachers, a networked library of documents, and their own simulation models in the pursuit of understanding. The article invokes the psychology of Piaget and Bruner and the educational models of Montessori and Suzuki while also laying out the technical details of how such a device could be constructed, and what would make it ideal for kids.

Figure 4.2: Cardboard mockup circa 1971–1972 (from Kay & Goldberg 1976)

Much of Kay's work in the early 1970s was devoted to creating the machine described in the 1972 article. By 1976, software prototypes had been in use with groups of children "of all ages" from the Palo Alto area for three or four years. In a 1976 report called *Personal Dynamic Media*, Kay and PARC colleague Adele Goldberg wrote:

> Aside from the potential future for education implied by getting kids to program, we realized that many of the problems involved in the design of a metamedium for creative thought, particularly those having to do with expressive communication, were brought strongly into focus when children down to the age of six were seriously considered as users.
>
> We felt then that the next time we tried to design a personal metamedium it should be done with children strongly in mind. We decided to expand our horizons to include studies into the nature of the learning and creative processes, visual and auditory perception, how to teach thinking, and how to show children the challenges and excitement of doing art and science. (Kay & Goldberg 1976, p. 9)

The scope implied by putting children at the centre of this "metamedium" drove the team's thinking to both enormous generalities and minute detail. In the early 1990s, Kay reflected on their evolving vision of personal computing:

> Not a personal dynamic vehicle, as in Englebart's metaphor opposed to the IBM "railroads," but something much more profound: a personal dynamic medium. With a vehicle one could wait until high school and give "drivers ed," but if it was a medium, it had to extend to the world of childhood. (Kay 1996*a*, p. 523)

Kay's focus on children, once established, has never wavered; despite numerous changes in direction—both within the Learning Research Group at Xerox PARC and in subsequent research and corporate contexts for his work—Kay has continued to design for children. But, unlike the clichés of children's software we have come to know—bright primary colours and cutesy icons—Kay's team took this task seriously enough to question first principles:

> Early on, this led to a 90-degree rotation of the purpose of the user interface from "access to functionality" to "environment in which users learn by doing." This new stance could now respond to the echos of Montessori and Dewey, particularly the former, and got me, on rereading Jerome Bruner, to think beyond the children's curriculum to a "curriculum of user interface."
>
> The particular aim of LRG was to find the equivalent of writing—that is, learning and thinking by doing in a medium—our new "pocket universe." (1996, p. 552)

Kay's ambition evidently ran on a pretty grand scale: not content simply to find educational applications for computers, Kay and his team self-consciously set out to redefine computing itself in educational terms. A child prodigy turned computer visionary, Kay was in his element; and if any environment were to prove fertile for such a wide-ranging undertaking it was the generously funded Xerox PARC of the 1970s, a research centre hosting the cream of American computer science and with little or no clear corporate mandate from Xerox itself. Kay's ego and passion had opportunity to stretch out in an unparalleled way. "[W]e were actually trying for a qualitative shift in belief structures," he wrote in 1996, "—a new Kuhnian paradigm in the same spirit as the invention of the printing press—and thus took highly extreme positions which almost forced these new styles to be invented" (p. 511). Kay's enthusiasm and unbridled romanticism is captured in his best-known quotation: "The

best way to predict the future is to invent it." Kay's future included children, personal computers, and a new kind of literacy.

"Late Binding" and Systems Design

But how would you do it? How would you set about to "invent" personal computing, almost from scratch, with very scant pre-existing work[6] to draw on in terms of working models or even research? A couple of possible strategies come to mind:

- You could survey your potential audience of 'users' to gather specific information about their needs, and then build a system so as to meet these needs most closely.
- Alternatively, you could develop a comprehensive vision of what personal computing 'should' be, develop a system according to these guidelines, and then work hard to train people to use it.

Neither of these approaches is foreign to us today; the first is known as "user-centred design," taking pains to compile lists of requirements and following up with extensive testing that the system indeed works the way the users want it to. The second, more hubristic method is unfortunately all too common, with the boundary between designers and users inscribed deeper with every painful training session. Both approaches assume that it is possible to know *in advance* what the system will be like; either by drawing from users or by innate knowledge.

There are no doubt elements of both of these in the early Dynabook project, but the more profound—and influential—note Kay struck in his early approach to inventing personal computing was one of *humility*. If the target audience for personal computing was adults—business people, for instance, as it was with the FLEX Machine—then a user-centered design approach or even brute-force training might have been plausible; Kay has a

6. I am, of course, stretching this for rhetorical purposes; Kay repeatedly points to the important precursors to his work: McCarthy's Lisp; Sutherland's Sketchpad; the JOSS system, Logo, RAND's Grail tablet interface; Englebart's augmentation project. But while these examples all contribute importantly to the idea of "personal" computing, none can claim to be a total personal computing environment in the sense that Kay's project aspired to be.

lot to say about prior examples of systems designed for particular user communities in his 1996 paper. But since his vision of personal computing put children at the centre, and because it was to be fundamentally *transformative*—in the same sense that the printing revolution was transformative in early modern Europe (Kay 1996*a*, p. 523; 2000*b*)—the starting point had to be one of acknowledged *ignorance*. How does one build a system that can grow into something yet unforeseen by either its users or its designers? Kay took three examples, or metaphors, for guidance.

The first metaphor was cell biology, which Kay had studied as an undergraduate in the early 1960s (on the heels of some of the most significant advances in the field and the identification of DNA in the 1950s).

> My biology major had focused on both cell metabolism and larger scale morphogenesis with its notions of simple mechanisms controlling complex processes and one kind of building block being able to differentiate into all needed building blocks. (1996*a*, p. 516)

> Instead of trying to build the complex artifacts from scratch—like trying to build living things cell by cell—many of the most important projects built a kernel that could *grow* the artifact as new knowledge was gained—that is: get one cell's DNA in good shape and let it help grow the whole system. (2004*a*)

Kay saw that in biological systems, there is a different order of mechanism at work than in the Newtonian model of the universe. "There's another kind of machinery that's very very different than the clockwork kind," he reflected (1996*b*). This insight is no doubt part of a large scale intellectual shift evident in the 20th century (and ongoing) toward a "systems view of the world" (Lazslo 1972; Prigogine & Stengers 1985). Kay points out several facts: the longest any atom resides in your body is 7 years; blood cells live less than 100 days; most of your body is less than two weeks old. We are composed of 100 trillion cells, 60 billion large information molecules, in constant motion, with pattern matches occuring every microsecond or so. We are not material so much as *patterns*: we stay alive by rebuilding the pattern, discarding the disarrayed stuff (2002*a*). This is a kind of mechanism that is qualitia-

tively different from any technology humankind has come up with, one capable of amazing things:

> A baby is able to get six inches longer about ten times in its life and you don't have to take it down for maintenance. [...] Imagine trying to make a 747 six inches longer! (1996*b*)

Kay's second guiding example is 'man-made': the United States Constitution,

> ...because the people who designed it realized that it would be very difficult to write laws for how people should live 50 years from their time and place, so they wisely made most of the Constitution a way of dealing with error situations and a way of keeping bad things from propagating. (Kay 1999)

> The Constitution itself is a set of principles for building a very complex dynamic structure that should last for centuries whose "parts" (that is, us!) come and go and are only somewhat intercooperative. (Kay 1995)

In his address to the *History of Programming Languages II* conference (1996*a*), Kay told his audience of computer scientists that the "best book on complex systems design" is Hamilton, Madison, and Jay's *The Federalist Papers*, which comprise the extended arguments supporting the (surprisingly concise) US Constitution's form and method, and which elaborate the whole idea of checks and balances, divisions and mixtures of powers, and mechanisms for amendment that are arguably the US Constitution's most important contributions (Hamilton et al. 1788/1987). Nowhere is the spirit of liberal humanism more clearly evident in the history of personal computing than here.

The third example is something much nearer to Kay's own experience: the development of the early ARPAnet, foundation of today's Internet. Kay witnessed first-hand, as a graduate student at Utah in the 1960s, the design and decision making processes behind it. Today, the Internet's status as a unique technological accomplishment is overshadowed by its immediate and more practical impacts in most of our lives, but, as Kay points out,

> the Internet has recycled all of its atoms and all of its bits probably twice now without being stopped—it's the only human artifact that has done that. (2002*a*)

What all this has to do with creating personal computing in the 1970s is this: one can't possibly anticipate in advance what the system is going to evolve into. Therefore, one's responsibility at the outset is to build in an open-ended fashion, to not commit oneself prematurely or to foreclose on possibilities by driving to a predefined end-goal; rather, to build for growth and for evolution. Note how close this is to the ARPA management strategy, to *fund directions, not goals*. Kay and Goldberg make the practical case thusly:

> The total range of possible users is so great that any attempt to specifically anticipate their needs in the design of the Dynabook would end in a disastrous feature-laden hodgepodge which would not be really suitable for anyone. We have taken an entirely different approach to this problem, one which involves the notion of providing many degrees of freedom and a way for any user to communicate his or her own wishes for a specific ability.
>
> Some mass items, such as cars and television sets, attempt to anticipate and provide for a variety of applications in a fairly inflexible way; those who wish to do something different will have to put in considerable effort. Other items, such as paper and clay, offer many dimensions of possibility and high resolution; these can be used in an unanticipated way by many, though tools need to be made or obtained to stir some of the medium's possibilities while constraining others.
>
> We would like the Dynabook to have the flexibility and generality of this second kind of item, combined with tools which have the power of the first kind. Thus a great deal of effort has been put into providing both endless possibilities and easy tool-making through a new *medium for communication* called Smalltalk. (Kay & Goldberg 1976, pp. 7–8 [italics added])

In computer science, there is a term which refers to this general idea, and it has become one of Kay's watchwords: *late binding*. Late binding, in the strict sense, means delaying the association of named entities in a program with actual computations (that is, with runtime objects) until the last minute for the sake of the dynamic interpretation of the program and its environment. Taken to extremes, the opposite concept would be "hard wired," in which there is no scope for timely interpretation of contexts. Note that most popular bogeyman images of computers correspond to the latter. Kay's usage of "late binding" is much broader, extending to the very architecture of the system being fluid:

[late binding] has some deeper and more profound properties that include abilities to actually change both the structure and metastructure of the language itself. Thus an important new idea can be assimilated into the constantly evolving process that is the system.

Another aspect of late-binding is the ability to change one's mind about already instantiated structures that are already doing work. These can be changed automatically on the fly without harming the work they are already doing. (Kay 2003*a*)

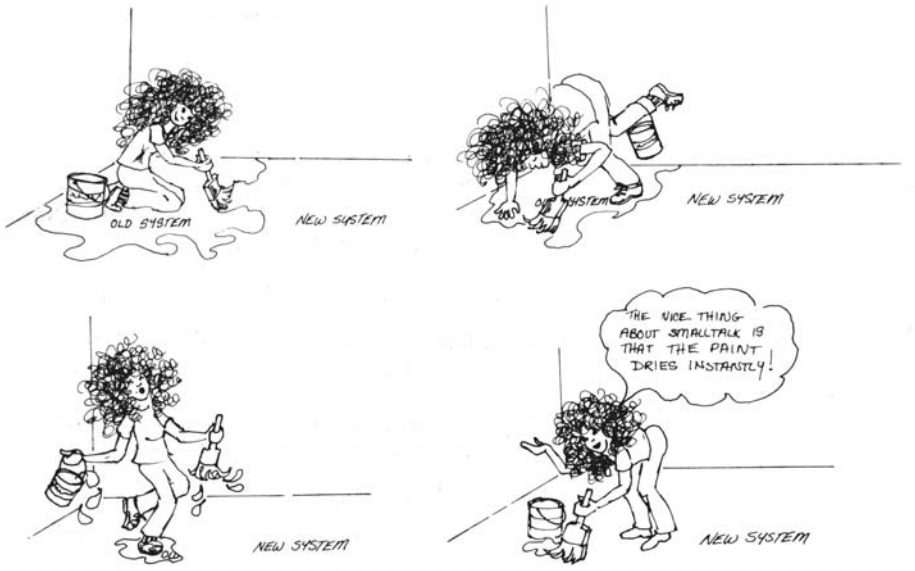


Figure 4.3: Cartoon by Ted Kaehler, from Goldberg & Robson (1983)

How different is this from the world we've come to inhabit? Our systems are a source of frustration, specifically because they are not fluid. Yet this fluidity was the cornerstone of Kay's strategy for building a personal computing paradigm for children. Implicated in this issue is the dynamic of *means* and *ends* and the rhetoric of instrumental rationality. Kay notes that adults, and businesses in particular, are especially prone to *instrumental rationality*—with the "judgement of any tool or idea solely in terms of the current goal structure of that person." The antidote, then, "the way to get ahead is to think about kids, not business. [...] Adults have way too much context—the enemy of real qualitative improvement" (Kay, 2003*a*).

By thinking of children as the target audience, and setting out to create a system with growth and evolution as its primary feature, Alan Kay and the Learning Research Group at Xerox PARC began to build their vision of personal computing.

Research Notes from Viewpoints Research Institute

From my trip to Viewpoints Research Institute (VPRI) in Glendale, CA, in April 2004. VPRI, a nonprofit organization, is the current home for Alan Kay's team and projects.

> *VPRI executive director Kim Rose tells me how she is an "early-binding" person working with the "late-binding" people. She says this without pause, but I stop her, and say, "You're not talking about software engineering here, are you?" She grins, glad that I got her drift, and says, "No, I mean that as a personality trait." Original Smalltalk designer Dan Ingalls, she continues, is a "late-binding" guy... he's coming to visit tomorrow, something we just found out today at about 3PM. Alan Kay, of course, is the patron saint of late binding, in both the traditional sense and Kim's metaphoric sense.*

Glendale, day 2:

> *Having spent several hours sifting through the VPRI filing cabinets, we talk about my use of these documents, many of which are lectures and talks Alan has given over the years. Kim defers to Alan, who welcomes me to take copies of whatever I like, but asks that I not directly quote anything marked "draft." But everything is marked "draft"! I observe that the "draft" claim is just another example of late binding; Alan acknowledges as much.*

Smalltalk—"A New Medium for Communications"

Smalltalk is without a doubt the most lasting of all of Alan Kay's technical contributions. Considering that a great number of now-ubiquitous features are attributed to him—from overlapping windows and pull-down menus to laptop and notebook computers—this is no small statement. Smalltalk can be seen as the still-evolving evolving embodiment of a deep vision of computing, a vision which has only partly been realized. Smalltalk is at once the core of Kay's educational vision and the core of his contribution to 'serious' computer science. It is also the crucible within which all the other inventions that make up personal computing came to their earliest fruition.

Just what Smalltalk *is* is difficult to capture in just a few words. It is, strictly speaking, a computer programming language. But more importantly, it is the embodiment of a self-consciously defined paradigm of computing (Kay repeatedly called it a "communications medium" rather than a programming language), one with profound historical implications for computer science and current software engineering practices, and one with even more profound—as yet largely unrealized—implications for literacy in a world interconnected by digital networks. Smalltalk has a substantial claim to being the original language for "object-

oriented programming" (OOP)[7], which is arguably the dominant professional programming paradigm today. Smalltalk's significance to personal computing, however, went right over most people's heads; when Apple Computer's Steve Jobs paid his famous visit to Xerox PARC in 1979, according to the unofficial "origin myth" of the Macintosh, he was shown three things: the graphical, overlapping-window interface; networked personal computers; and Smalltalk.

> And they showed me really three things. But I was so blinded by the first one I didn't even really see the other two. (from Cringley 1996: "Triumph of the Nerds, Part 3," PBS)

In the mid 1980s, Apple began to sell computers with graphical user interfaces. By the mid 1990s, networked personal computers were becoming commonplace. But Smalltalk and the paradigm it represents remains a behind-the-scenes component of modern computing, despite Kay's ambitions for the Dynabook. Far from just a software engineering tool, Smalltalk was to be "the exemplar of the new computing, in part, because we were actually trying for a qualitative shift in belief structures—a new Kuhnian paradigm in the same spirit as the invention of the printing press" (Kay 1996*a*, p. 511).

Objects and messages

The Smalltalk language more or less defined the "object-oriented paradigm," as it was the first language system to fully embrace the idea of message-passing objects as the basis for software.

> New ideas go through stages of acceptance, both from within and without. From within, the sequence moves from "barely seeing" a pattern several times, then noting it but not perceiving its "cosmic" significance, then using it operationally in several areas; then comes a "grand rotation" in which the pattern becomes the center of a new way of thinking, and finally, it turns into the same kind of inflexible religion that it originally broke away from. (Kay 1996*a*, p. 514)

7. Purists insist that Ole-Johan Dahl and Kristin Nygaard's *Simula* language, dating from the mid 1960s, is the original OOP language. However, Simula was an attempt to add object-oriented ideas to an existing procedural language (Algol), while Smalltalk was designed from the ground up around the idea of objects and message-passing.

As a new graduate student at the University of Utah in 1966, Kay was exposed to two systems that carried the seeds of object-oriented thinking: the first was Ivan Sutherland's *Sketchpad*, the first interactive drawing system; the second was *Simula*, a Norwegian adaptation of the Algol programming language for creating simulations. Both systems had employed a "master" and "instance" architecture, with instance-objects inheriting both attributes and behaviour from the master-objects they are derived from. For Kay, this idea resonated with a number of other ideas he had encountered:

> This was the big hit, and I have not been the same since. I think the reasons the hit had such impact was that I had seen the idea enough times in enough different forms that the final recognition was in such general terms to have the quality of an epiphany. My math major had centered on abstract algebras with their few operations applying to many structures. My biology major had focused on both cell metabolism and larger scale morphogenesis with its notions of simple mechanisms controlling complex processes and one kind of building block being able to differentiate into all needed building blocks. The 220 file system, the B5000, Sketchpad, and finally Simula, all used the same idea for different purposes. Bob Barton, the main designer of the B5000 and a professor at Utah, had said in one of his talks a few days earlier, "The basic principle of recursive design is to make the parts have the same power as the whole." For the first time I thought of the whole as the entire computer and wondered why anyone would want to divide it up into weaker things called data structures and procedures. Why not divide it up into little computers, as time sharing was starting to? But not in dozens. Why not thousands of them, each simulating a useful structure? (p. 516)

Kay's graduate work was an exercise in fleshing out this idea—which ultimately came to be known as *object orientation*—in a workable system. Following his meeting with Seymour Papert and seeing a possible future with millions of personal computers run by children—Kay's work at the new Xerox PARC focused on the design of an extensible, late-binding system based on objects and message-passing. Smalltalk grew out of Kay's early work on the FLEX Machine, and began to be fleshed out in the "Dynabook" vision. As a language and computing environment, Smalltalk would share much with Logo: it would be straightfor-

ward and simple and allow a user to define her own language structures. But Smalltalk was founded on a different set of founding principles than Logo (which was, historically, "Lisp for kids"). Smalltalk's promise was that objects would make computing "child's play".

The Design of Smalltalk

Kay and the Learning Research Group at PARC set about trying to identify just what Smalltalk's founding priciples properly were, "during this state of grace (before any workable implementation) ... trying to understand what 'beautiful' might mean with reference to object-oriented design" (1996*a*, p. 529). Smalltalk went from pencil-and-paper to working implementation by 1972, by which time Kay had distilled his idea into six foundational premises:

- Everything is an *object*.
- Objects communicate by sending and receiving *messages* (in terms of objects)
- Objects have their *own memory* (in terms of objects)
- Every object is an *instance* of a *class*[8] (which must be an object)
- The class holds the shared *behaviour* for its instances (in the form of objects in a program list)
- To evaluate a program list, control is passed to the first object and the remainder is treated as its message. (Kay 1996*a*, p. 534; see also Schoch 1979, p. 64)

Smalltalk's LISP Legacy

Logo was a direct derivative of LISP. While the syntax of the two languages differs—Logo was designed to be simple for young children to pick up quickly—the fundamental functional architecture is the same.

Kay had enormous admiration for Lisp, which he got to know intimately during a brief stint at the Stanford AI Lab (SAIL) in 1969–1970, where John McCarthy had been based. Kay "could hardly believe how beautiful and wonderful the idea of LISP was," but had serious criticisms of the actual language, which he felt compromised the fundamental idea at the language's core (Kay 1996a, p. 525). This critique was part of Kay's inspiration with Smalltalk.

> *...this started a line of thought that said "take the hardest and most profound thing you need to do, make it great, and then build every easier thing out of it" [...] needed was a better "hardest and most profound" thing. Objects should be it. (p. 525)*

Kay's claim that the "most powerful language in the world" could be written in "a page of code" was directly inspired by LISP, which could be described/implemented *in itself* in about a page. Kay took this as a design goal for early Smalltalk.

8. The idea of classes and instances derived from them has very definite Platonic roots. John Schoch, one of the LRG team, wrote of the relationship of classes and Platonic forms in his paper on Smalltalk-72 (Schoch 1979, p. 70–71).

These six main ideas are the result of Kay's attempt to find the underlying factors that make a system both simple and capable of great complexity:

> A fertilized egg that can transform itself into the myriad of specifications needed to make a complex organism has parsimony, generality, enlightenment, and finesse—in short, beauty, and a beauty much more in line with my own esthetics. I mean by this that Nature is wonderful at both elegance and practicality—the cell membrane is partly there to allow useful evolutionary kludges to do their necessary work and still be able to act as components by presenting a uniform interface to the world. (1996*a*, p. 530)

The cell and cell membrane are key metaphors—analogous to the "black boxes" of Actor-Network Theory—in understanding what Kay means by object-orientation, but they are also somewhat misleading. Our temptation in looking at systems of almost any sort is to focus on the *entities* that we see, rather than on the relationships, interactions, and transformations that occur among them. So it is with the noun-centric Actor-Network Theory, as opposed to the more process-oriented "sociology of translation." Kay has repeatedly expressed his regret that he chose the term "object-oriented" instead of the more relational concept of "message-oriented." What is important about biological cells in Kay's systems-theory rendering of them isn't what they're made of, but rather their modes of interacting. The cell membrane is here read as a black-boxing mechanism in precisely Latour's sense; it means we can ignore the inner workings of the cell and focus our attention instead on the *interface* that the cell presents as a means of creating larger structures (like tissues and organs). Kay's point is that the building blocks can be—should be—transcended in a sense, putting the emphasis rather on the messages and interactions that in turn generate the many layerings of structure of complex systems. This shift of emphasis is, to my mind, akin to the semiotic shift described in Chapter 3 above: when the components of a system—alphabetic glyphs, gears, bits—reach a level of simplicity and standardization, the production of meaning no longer accrues to the individual component, but to their larger arrangement; we do not interpret the individual letters of the Roman alphabet as the Egyptians did with their hieroglyphics; we look instead for meaning in the larger literary

structures of which they are built. This same theme is apparent in Kay's systems design exemplars: it is not the cell itself but its interrelations which gives rise to complex biological systems; the US Constitution does not itself dictate anything about how an individual should behave, but rather sets up a process for how individuals can organize their interrelations; the ARPAnet, like the Internet, dictates nothing about the characterisitics of connected systems; it only provides a means for their intercommunication.

Late binding in Smalltalk

The Smalltalk 'origin myth' involves a bet Kay made with LRG colleagues Dan Ingalls and Ted Kaehler that "the most powerful language in the world" could be described in "a page of code" (1996*a*, p. 533). Ingalls and Kaehler reportedly called his bluff and so Kay spent the next several weeks working between 4 and 8AM each day on the bet. The result was the design for Smalltalk-72, the first working implementation of the language. In order to fit the entire design for a powerful programming language in less than one page (McCarthy's Lisp was describable in less), Kay's challenge was to define only the absolute foundational structures from which everything else can be grown. Such a language design has no 'features' as such; it is an exercise in the utmost parsimony and abstraction.

Process vs. State

In language inspired by A. N. Whitehead's "process philosophy" (which interestingly also appears in Latour's more recent writings), Kay wrote:

> *The basic idea is to exploit the duality between functions and tables (or processes and memory). English has nouns which refer to "objects," and verbs which refer to "actors" and "relators." This is a Newtonian epistemology. Modern physics and philosophy tend toward the idea that both "objects" and "actors" are just different aspects of the notion of process. A process has state (a set of relations having only to do with it) which changes as time (defined as interactions with other objects) passes. Using this view, "data" is a process which changes "slowly," "function" is a process which changes more rapidly. Each process has the logical attributes of a complete "micro" computer: they can have inputs, give back outputs, act as a memory on file system, perform computations, be interrupted, etc. Since a "computer" can emulate all other computers (modulo time and space), having a notion of a process in a language allows useful ideas such as arrays, records, recursive procedures, etc. to be added to the repertoire at any time. (Kay 1972)*

To go from a single page description of the foundations of Smalltalk to a working implementation and then to the first prototypes of a graphical user interface (such as we all now use on our personal computers), the first paint program, structured document editing, music capture and editing, animation, and the

LRG's research with children as end-users—all of which emerged with the Smalltalk-72 system—is testament to the power of a late-binding system. All of these 'features' were created in the Smalltalk-72 environment once it was up and running (after Dan Ingalls had implemented the basic design, Kay wrote, "there was nothing to do but keep going.") The core of the research conducted with kids was towards playing with and extending the tools and 'features' in the environment. Kay's team at PARC worked for 4 more years on the Smalltalk-72 environment, on Xerox' *Alto* minicomputers—which Kay's team called "interim dynabooks."[9]

Smalltalk-72 was ultimately overhauled for a variety of reasons. One of which was that it was in a sense *too* late-binding. The original design had each object in charge of defining the syntax of messages passed to it. For instance, a *number* object would specify how arithmetic operations should be called, and a *text-string* object would define how its operations would be called, and there was no necessary parallel or congruity between these. As users defined a larger and larger superstructure of tools and applications over time, the ways in which the Smalltalk system worked grew more varied and complex. This fit well with the ideal of an individual user gradually building the system to her own liking, but it made collaborative projects difficult—the way one user had constructed something might not make sense to another user (Goldberg & Ross 1981), and it was difficult to grow programs to large scale and complexity.

The substantial result is that Dan Ingalls re-built Smalltalk again to create *Smalltalk-76*, which featured a much more consistent approach to the design; everything in Smalltalk-76 adhered very closely to the six main ideas outlined above: everything was an object, and the notion of objects being instances of classes which defined their behaviour led to a class *hierarchy*—which has become one of the fundamentals of object-oriented programming. The classic example (from Ingalls 1978) is this:

9. The Alto, designed and constructed by PARC's Computer Science Lab, is often described as the first "personal computer." Far from the portable laptops of Kay's imagination, the Altos were the size of bar fridges, and sat under a desk, generating a lot of heat. Nevertheless, they were constructed in the thousands in the 1970s and were used as personal workstations by staff at PARC. (Hiltzik 1999)

1. A *rectangle* object knows how to display itself onscreen, in response to messages specifying its size and position;
2. A *window* object is a kind of *rectangle* that acts as a frame for interactions and content. It receives messages generated by the movement of the mouse and its button clicks, if the pointer is within the window's boundaries;
3. A *document window* object is a kind of *window* which is designed to hold some text. It can respond to mouse-click messages as positioning the insertion point or selecting text.

This kind of hierarchical structure led to what Adele Goldberg called "incremental design" (1979): that a user can create tools and media objects by taking pre-existing building blocks and then creating new versions of them that have added functionality or features (by "subclassing"—that is, specializing the behaviour of object classes in new classes derived from them). The message-sending syntax in Smalltalk-76 was made more consistent, which led to more readable programs (one "learns to write by reading, and to read by writing," wrote Goldberg in 1998), which, in turn, led to a greater emphasis on learners as *designers*, rather than just *tinkerers*—they were able to create more complex and multi-layered constructions.

Smalltalk went through at least one more substantial revision, culminating in the *Smalltalk-80* that was released to the world beyond Xerox PARC in the early 1980s (a series of articles in *BYTE*, August 1981; Goldberg & Robson 1983). But it is important to bear in mind Kay's

Reflections on "Epistemological Pluralism"

Kay and Goldberg's notion of learners as designers is an intriguing alternative to the two major "cognitive styles" identified in Turkle and Papert's (1991) article, "Epistemological Pluralism and the Revaluation of the Concrete" which opposed "hard" (top down, structured, male, engineering) from "soft" (bottom-up, exploratory, female, bricolage) styles. Kay's rendering is somewhat more nuanced:

> *There are two basic approaches to personal computing. The first one, which is analogous to musical improvisation, is exploratory: effects are caused in order to see what they are like and then tracked down, understood, and fixed. The second, which resembles musical composition, calls for a great deal more planning, generality and structure. The same language is used for both methods but the framework is quite different.*
>
> *From our study we have learned the importance of a balance between free exploration and developed curriculum. The personal computing experience is similar to the introduction of a piano into a third-grade classroom. The children will make noise and even music by experimentation, but eventually they will need help in dealing with the instrument in non-obvious ways. (Kay 1977, p. 13)*

repeated admonition that, true to its very design, Smalltalk isn't ever *finished* but continues to be a vehicle for getting to the next place—a new dynamic media environment for children. By this treatment, Smalltalk isn't thus a better language to teach programming *qua* programming; it is a comprehensive model for late binding and for complete end-user control. Thus, Smalltalk's ultimate and ongoing goal, Kay suggests, is to transcend itself.

The Smalltalk environment

As Smalltalk evolved, especially after the 1976 design, the Smalltalk environment seems to have become more important than the language *per se*. This was, in a sense, what Kay was after; the language itself was to merely be a vehicle for getting to a place where one could build one's own tools and media to suit the task at hand, a "language for building languages." Adele Goldberg described Smalltalk-76 as "a basis for implementing and studying various user-interface concepts" (Goldberg & Ross 1981). Smalltalk's contributions to user-interface concepts are undoubtedly its most widely-known facet—this is what Steve Jobs saw when he visited Xerox PARC in 1979 (Goldberg 1998, p. 69–70). But Smalltalk's intent was not merely to make computers "user-friendly." Kay wrote:

> I felt that because the content of personal computing was interactive tools, the content of this new authoring literacy should be the creation of interactive tools by the children. (1996*a*, p. 544)

The creation of interactive tools positions the computer as a *media* device. Kay's ARPA background significantly framed the notion of computers as *communications media* rather than *calculating machines*, but Smalltalk's orientation meant that "objects mean multimedia documents; you almost get them for free. Early on we realized than in such a document, each component object should handle its own editing chores" (p. 538). You get them "for free" because there isn't a foundational divide between *data structures* and *procedures*, as in conventional programming models; once this conceptual limitation has been escaped, it is as straightforward to define objects that manipulate audio samples as the ones that manipulate text. As early as Smalltalk-72, the LRG team were experimenting with bitmap painting

programs and animation, music sequencing and editing, and WYSIWYG document editing. The now-commonplace idiom of "authoring" and the toolbar motif that we find in our word processors and graphics applications were pioneered here.

Kay was struck by the computer's power to represent any other media, be it textual, graphical, or aural. Kay and Goldberg wrote,

> Every message is, in one sense or another, a *simulation* of some idea. It may be representational or abstract, isolated or in context, static or dynamic. The essence of a medium is very much dependent on the way messages are embedded, changed, and viewed. Although digital computers were originally designed to do arithmetic computation, the ability to simulate the details of any descriptive model means that the computer, viewed as a medium itself, can be *all other media* if the embedding and viewing methods are sufficiently well provided. Moreover, this new "metamedium" is *active*—it can respond to queries and experiments—so that the messages may involve the learner in a two-way conversation. This property has never been available before except through the medium of an individual teacher. We think the implications are vast and compelling. (1976, p. 4)

These foundational principles, and their realization in the evolving forms of Smalltalk that emerged during the 1970s, begin to define what Kay has called (2004*a*) the "PARC genre" of personal computing, from which we have inherited certain elements today: authoring tools, direct manipulation interfaces, exploratory systems, and so on. What we have not inherited is the *ethic* of mutability, wherein we would assume that every component of a system is open to be explored, investigated, modified, or built upon. The Smalltalk systems "crystallized a style of programming" (Kay 1978) in which the entire system, top to bottom, was open to the user's exploration and interaction, in which the line between tool and medium was deliberately blurred.

"Doing With Images Makes Symbols"

Seymour Papert's research with children had drawn heavily on the theories of developmental psychologist Jean Piaget, with whom he worked in the late 1950s and early 1960s. Piaget's

developmental theory—which holds that children's cognition passes through four normal stages—sensorimotor, preoperational, concrete operational, and formal operational—has two important features with special relevance to Papert's work. The first is Piaget's conception of the child as *epistemologist*, actively building an understanding of the world and then reflecting upon that understanding (and, ultimately, on the process of building it). In his rememberance of Piaget in *Time's* 1999 issue on "The Century's Greatest Minds," Papert wrote:

> Children have real understanding only of that which they invent themselves, and each time that we try to teach them something too quickly, we keep them from reinventing it themselves. (Papert 1999)

There is more here than a simple rendering of constructivism; it speaks instead to Piaget's (and Papert's) concern that the business of 'teaching' children about the world may run counter to their ability to make sense of it for themselves, if in teaching we attempt to overlay a way of seeing the world more appropriate to adults. Piaget's insistence was to take seriously the idea that through the various developmental stages, children's ways of thinking (or "ways of knowing") may be very different from adults'—not wrong or lacking, but different. Piaget thus saw children's thinking, children's epistemology, as intellectually interesting in itself.

The second feature of Piaget's theory of importance to Papert's work is the distinction between *concrete* and *formal* thinking which underlies Piaget's third and fourth stages. That there is a well-documented phenomenon wherein children cannot perform certain sorting and combining tasks (and which Piaget's research uses to differentiate these stages) does not necessarily imply that young children cannot approach this kind of thinking. Rather, Papert argued, with the right kind of representational apparatus, they may; hence Logo. This is an application of the epistemological insight above, and perhaps the best way to express it is to note, as Alan Kay did:

> ...that young children are not well equipped to do "standard" symbolic mathematics until the age of 11 or 12, but that even very young children can do other kinds of math, even advanced math such as topology and differential geometry, when it is presented in a form that is well matched to their current thinking processes. The Logo turtle with its local coordinate system (like the child, it is always at the center of its universe) became a highly successful "microworld" for exploring ideas in differential geometry. (1990, p. 194)

The computer, in Papert's conception, isn't a *bridge* to the formal operational stage so much as it is a means to approach the ideas inherent in mathematics while still remaining within the cognitive repertoire of the concrete stage—through the computer-mediated introduction of a "body-syntonic mathematics" (Papert, 1980a, p. 205). *This* is what computers are good for, according to Papert. In a computer-rich culture, Papert wrote, "Children may learn to be systematic before they learn to be quantitative!"

Kay worked forward from Papert's conception, but moved in a slightly different direction. The key idea for Kay was that children could use computing to gain a different sort of handle on the world. Kay, following McLuhan, felt that new media come with new ways of thinking about the world. The relationship to Piaget *per se* was downplayed in favour of the Anglo-American developmental psychologist Jerome Bruner, who in the 1960s postulated three mentalities: *enactive*, *iconic*, and *symbolic*, and, rather than being strictly developmental (i.e., stage-ordered), these mentalities developed in response to environmental factors (Bruner 1966, p. 10ff).

> The work of Papert convinced me that whatever interface design might be, it was solidly intertwined with learning. Bruner convinced me that learning takes place best environmentally and roughly in stage order—it is best to learn something kinesthetically, then iconically, and finally the intuitive knowledge will be in place to allow the more powerful but less vivid symbolic processes to work at their strongest. This led me over the years to the pioneers of environmental learning: Montessori Method, Suzuki Violin, and Tim Gallwey's Inner Game of Tennis, to name just a few. (Kay 1990, p. 195)

The word "interface" here is the key one; prior to the 1970s, "user interface" had a different meaning. Kay's breakthrough idea—though he would no doubt say that he was drawing on precursors from the 1960s (Sketchpad, GRAIL,[10] and Logo)—was that 'computing' could and should be operating on more than just the symbolic, abstract level; that it can and should have enactive (kinesthetic) and iconic (image-based) modalities, since computers were *media,* and not just tools. But the importance of enabling the enactive and iconic mentalities in computing is not merely to allow computers to be accessible to children in one or another cognitive 'stage,' but because, Kay said, "no single mentality offers a complete answer to the entire range of thinking and problem solving. User interface design should integrate them at least as well as Bruner did in his spiral curriculum ideas" (Kay 1990, p. 195). The goal was not to make computers available to children, but "to find the equivalent of writing—that is, learning and thinking by doing in a medium—our new 'pocket universe'" (1996*a*, p. 552). The important thing about language and reading and writing, Kay noted, is that very young children use the same English language as poets, scientists, and scholars; what differs isn't the language, but the context and the sophistication of the user.

But as Piaget's and Bruner's work underscores, the relative sophistication of these users is not simply a matter of degree, but of style. Kay read a study by the French mathematician Jacques Hadamard (1954) on the personal reflections of over 100 leading mathematicians and scientists; Hadamard found that very few of them reported thinking about mathematics in a "symbolic" way (i.e. the way it is written, and the way it is taught in school); most reported thinking in terms of *imagery*, and a significant number (including Albert Einstein) reported that their sensibilities about mathematics had a *kinesthetic* basis.

Kay wrote that Hadamard's study indicates "strongly that creativity in these areas is not at all linked to the symbolic mentality (as most theories of learning suppose), but that the important work in creative areas is done in the initial two mentalities—most in the iconic (or figurative) and quite a bit in the enactive" (1990, p. 195). Of course mathematics as a

10. GRAIL was an early but sophisticated pen-and-tablet graphics system developed at the RAND Corporation in the late 1960s.

'language' is the very model of abstract, symbolic representation, and undoubtedly this is essential to the communication of its ideas; but this does not imply that mathematicians themselves necessarily think this way. Bruner's enactive, iconic, and symbolic are to be thought of as mentalities rather than hierarchical stages; they may develop in sequence, but we do not move 'out' of the earlier stages.[11] That is to say, what is 'developmental' about the three mentalities may be developmental only by virtue of the contexts of exposure and use. Kay sums this up nicely:

> The world of the symbolic can be dealt with effectively only when the repetitious aggregation of concrete instances becomes boring enough to motivate exchanging them for a single abstract insight. (1984, p. 6)

The implication for Kay's evolving notion of user interface is that "no single mentality offers a complete answer to the entire range of thinking and problem solving" (Kay 1990, p. 195). The Smalltalk environment, thus, had to be a system which could support interaction in all three modes, and its development over three decades can be seen as a progression towards a greater embodiment of this thinking. The guidelight for this development was Kay's neat synthesis of Bruner's schema: *"doing with images makes symbols."* User of computer media, therefore, should be *doing* something, be it pointing, moving, or manipulating objects on screen; those objects should have a visual, or iconic representation. This has an obvious developmental application: users can begin by manipulating concrete representations (c.f. Montessori's use of manipulatives) and gradually build the familiarity which allows more abstract, symbolic modalities.

11. Goldman-Segall's "epistemological attitudes" (1998, p. 244ff) is a related thought, itself in reaction to Papert & Turkle's 'hard' and 'soft' styles. That epistemological attitudes are conceived as "frames" puts a more contextual and dialogical light on it, rather than innate (or even learned, but stable) qualities.

Ways of Knowing: Narrative, Argumentation, Systems Thinking

Kay's insight at the end of the 1960s was that a new age of personal computing was on the horizon, in which

> ...millions of potential users meant that the user interface would have to become a learning environment along the lines of Montessori and Bruner [...] early on, this led to a 90-degree rotation of the purpose of the user interface from "access to functionality" to "environment in which users learn by doing." This new stance could now respond to the echos of Montessori and Dewey, particularly the former, and got me, on rereading Jerome Bruner, to think beyond the children's curriculum to a "curriculum of user interface." (1996*a*, p. 552)

What is a *curriculum of user interface?* Perhaps the best way to answer this is to look to another taken-for-granted medium—the printed book—and try to draw an analogy. As Marshall McLuhan has eloquently shown—and scholars such as Jack Goody, Walter Ong, and Elizabeth Eisenstein have elaborated—alphabetic literacy of the kind nurtured by the printing revolution of the Early Modern period has conditioned our thinking so deeply that we can barely imagine what it might have been otherwise. McLuhan's insights into alphabetic culture underscore the notion of an *alphabetic curriculum* that has been keystone of Western education in modern times to be sure, and really since the Greeks developed the alphabet in the modern sense: we do not merely *read* by way of alphabetic constructs, we organize our very being according to the kinds of logic prescribed by alphabetic literacy.

Alphabetic literacy as pioneered by the classical Greeks and especially print literacy as it appeared in the early 17th century has had profound implications for how we know the world; how we represent it, and the kinds of assumptions we make about it. One of the most profound implications is the development of alternatives to narrative expression. Kay pays special attention to this dynamic in the development of modernity. Of narrative and narrative-based ways of understanding the world, Kay writes:

> When one proverb contradicts another, it doesn't matter—just as it doesn't matter that the movie you liked last night contradicts the movie you liked last week. The important thing about stories is how good they are right now. Stories happen in the here and now; they create their own environment. Even when they purport to be about general knowledge, what really matters is how well they satisfy the listener. (Kay 1997, p. 17)

But we do not represent everything in narrative form. Since the early 17th century, Kay argues, more and more of the most influential cultural expressions in Western society have taken non-narrative forms:

> If we look back over the last 400 years to ponder what ideas have caused the greatest changes in human society and have ushered in our modern era of democracy, science, technology and health care, it may come as a bit of a shock to realize that none of these is in story form! Newton's treatise on the laws of motion, the force of gravity, and the behaviour of the planets is set up as a sequence of arguments that imitate Euclid's books on geometry. (Kay 1995)

The most important ideas in modern Western culture in the past few hundred years, Kay claims, are the ones driven by argumentation, by chains of logical assertions that have not been and cannot be straightforwardly represented in narrative. Historians Hobart & Schiffman identify the 16th-century French thinker François Viète, the developer of generalized algebra and geometry, as the turning point to modern analytic thought—the more famous Rene Descartes having built substantially on Viète's foundations (Hobart & Schiffman 1998, p. 123ff).

But more recent still are forms of argumentation that defy linear representation at all: 'complex' systems, dynamic models, ecological relationships of interacting parts. These can be hinted at with logical or mathematical representations, but in order to flesh them out effectively, they need to be *dynamically modeled*. This kind of modelling is in many cases only possible once we have computational systems at our disposal, and in fact with the advent of computational media, complex systems modeling has been an area of growing research, precisely because it allows for the representation (and thus conception) of knowledge beyond what was previously possible. In her discussion of the "regime of computation" inherent in the work of thinkers like Stephen Wolfram, Edward Fredkin, and Harold Morowitz, N. Katherine Hayles explains:

John Conway and A-Life

In 1970, Martin Gardner published a famous column in *Scientific American* celebrating a simple solitaire game of population dynamics created by British mathematician John Conway. The game was called Life. It is, very simply, a set of very simple algorithms that produce complex behaviour in *cellular automata*. The study of cellular automata pre-exists Conway's simulation, and has developed into a rapidly growing and complex branch of mathematics (Stephen Wolfram's 1200-page *A New Kind of Science* was a popular bestseller in 2002). But Life was simple enough to be implemented in a variety of simple forms: the small number of algorithms meant that the simulation could be run by hand on a checkerboard; novice programmers could easily produce Life as software in a few lines of code (I recall writing one in BASIC when I was an undergraduate).

The important thing about Life is that it demonstrates powerfully how very simple systems can produce complex behaviour, and that the business of modelling them, mathematically, computationally, is trivially easy.

> Whatever their limitations, these researchers fully understand that linear causal explanations are limited in scope and that multicausal complex systems require other modes of modeling and explanation. This seems to me a seminal insight that, despite three decades of work in chaos theory, complex systems, and simulation modeling, remains underappreciated and undertheorized in the physical sciences, and even more so in the social sciences and humanities. (Hayles 2005, p. 30)

Kay's lament too is that though these non-narrative forms of communication and understanding—both in the linear and complex varieties—are key to our modern world, a tiny fraction of people in Western society are actually fluent in them.

> In order to be be completely enfranchised in the 21st century, it will be very important for children to become fluent in all three of the central forms of thinking that are now in use. [...] the question is: How can we get children to explore ways of thinking beyond the one they're "wired for" (storytelling)[12] and venture out into intellectual territory that needs to be discovered anew by every thinking person: logic and systems "eco-logic?" (1996*c*)
>
> We can learn many things as children in a village culture. We can learn how to live our lives successfully. We can learn what the culture believes. We can learn how to hunt and fish and farm. We can learn a lot of things simply by watching others. But school was invented so people could learn the hard things, the things we don't learn naturally. School was invented so we could learn the created things that actually require us to change what's inside our heads, to learn what Seymour Papert calls powerful ideas. (Kay 1997, p. 18)

In this we get Kay's argument for 'what computers are good for' (not to mention a particular notion of what schools might be good for). It does not contradict Papert's vision of children's access to mathematical thinking; rather, it generalizes the principle, by applying Kay's vision of the computer as medium, and even *metamedium*, capable of "simulating the details of any descriptive model." The computer was already revolutionizing how science is done, but not general ways of thinking. Kay saw this as the promise of personal computing, with millions of users and millions of machines.

> The thing that jumped into my head was that simulation would be the basis for this new argument. [...] If you're going to talk about something really complex, a simulation is a more effective way of making your claim than, say, just a mathematical equation. If, for example, you're talking about an epidemic, you can make claims in an essay, and you can put mathematical equations in there. Still, it is really difficult for your reader to understand what you're actually talking about and to work out the ramifications. But it is very different if you can supply a model of your claim in the form of a working simulation, something that can be examined, and also can be changed. (2002*b*)

The computer is thus to be seen as a modelling tool. The models might be relatively mundane—our familiar word processors and painting programs define one end of the

12. Kay is presumably drawing on Bruner's notion of a foundational "narrative construal of reality" (Bruner 1991; 1996)

scale—or they might be considerably more complex. It is important to keep in mind that this conception of computing is in the first instance *personal*—"personal dynamic media"—so that the ideal isn't simulation and modelling on some institutional or centralized basis, but rather the kind of thing that individuals would engage in, in the same way in which individuals read and write for their own edification and practical reasons. This is what defines Kay's vision of a literacy that encompasses logic and systems thinking as well as narrative.

And, as with Papert's enactive mathematics, this vision seeks to make the understanding of complex systems something to which young children could realistically aspire, or that school curricula could incorporate. Note how different this is from having a 'computer-science' or an 'information technology' curriculum; what Kay is describing is more like a systems-science curriculum that happens to use computers as core tools:

> So, I think giving children a way of attacking complexity, even though for them complexity may be having a hundred simultaneously executing objects—which I think is enough complexity for anybody—gets them into that space in thinking about things that I think is more interesting than just simple input/output mechanisms. (1996*a*, p. 597)

What is Literacy?

> *The music is not in the piano. – Alan Kay*

The past three or four decades are littered with attempts to define "computer literacy" or something like it. I think that, in the best cases, at least, most of these have been attempts to establish some sort of conceptual clarity on what is good and worthwhile about computing. But none of them have won large numbers of supporters across the board.

Kay's appeal to the historical evolution of what literacy has meant over the past few hundred years is, I think, a much more fruitful framing. His argument is thus not for computer literacy *per se*, but for *systems literacy*, of which computing is a key part. In a (2002*a*) lecture, Kay said, "Every idea, no matter how revolutionary it may appear, is built on previous ideas. … What interests me … is adding something more to literacy. And this is a

grand tradition." Drawing a profound example from the history of literacy in Europe, Kay wrote in 1998 that

> ...we'll know if we have the first Dynabook if we can make the end-user experience one of "reading and writing" about "powerful ideas" in a dynamic form, and to do this in such a way that large percentages of the bell-curve can learn how to do this. When Martin Luther was in jail and contemplating how to get the Bible directly to the "end-users" he first thought about what it would take to teach Latin to most Germans. Then he thought about the problems of translating the Bible to German. Both were difficult prospects: the latter because Germany was a collection of provinces with regional dialects, and the dialects were mostly set up for village transactions and court intrigues. Interestingly, Luther chose to "fix up" German by restructuring it to be able to handle philosophical and religious discourse. He reasoned that it would be easier to start with something that was somewhat familiar to Germans who could then be elevated, as opposed to starting with the very different and unfamiliar form of Latin. (Not the least consideration here is that Latin was seen as the language of those in power and with education, and would partly seem unattainable to many e.g. farmers, etc.) (Kay 1998*a*)

That this is a massive undertaking is clear in the Luther example, and the size of the challenge is not lost on Kay. Reflecting on the difficulties they faced in trying to teach programming to children at PARC in the 1970s, he wrote that

> the connection to literacy was painfully clear. It is not just enough to learn to read and write. There is also a literature that renders ideas. Language is used to read and write about them, but at some point the organization of ideas starts to dominate the mere language abilities. And it helps greatly to have some powerful ideas under one's belt to better acquire more powerful ideas (Kay 1996*a*, p. 545).

Because literacy is about ideas, Kay connects the notion of literacy firmly to *literature*:

> What is literature about? Literature is a conversation in writing about important ideas. That's why Euclid's *Elements* and Newton's *Principia Mathematica* are as much a part of the Western world's tradition of great books as Plato's

> Dialogues. But somehow we've come to think of science and mathematics as being apart from literature. (2002*b*)

There are echoes here of Papert's lament about "mathophobia"—not fear of math but the fear of learning (Papert 1980, pp. 38–40) that underlies C.P. Snow's "two cultures," and which surely underlies our society's love-hate relationship with computing. Kay's warning that too few of us are truly fluent with the ways of thinking that have shaped the modern world—logical argument and systems dynamics—finds an anchor here. How is it that Euclid and Newton, to take Kay's favourite examples, are not part of the canon, unless one's very particular scholarly path leads there? We might argue that we all inherit Euclid's and Newton's ideas, but in distilled form. But this misses something important, and *I know I've missed something important* in my understanding of math and science. Kay makes this point with respect to Papert's experiences with Logo in classrooms:

> Despite many compelling presentations and demonstrations of Logo, elementary school teachers had little or no idea what calculus was or how to go about teaching real mathematics to children in a way that illuminates how we think about mathematics and how mathematics relates to the real world. (1997, p. 19)

The problem, in Kay's portrayal, isn't "computer literacy," it's a larger one of familiarity and fluency with the deeper intellectual content; not just that which is specific to math and science curriculum. Kay's diagnosis runs very close to Neil Postman's critiques of television and mass media (Postman was a member of the advisory board for the Viewpoints Research Institute until his death in 2003); that we as a society have become incapable of dealing with complex issues. Postman charges that public argument on the scale of that published in and around the US Constitution would be impossible today, because the length and depth of the argumentation simply would not fit in a television format, newspapers would not print it, and too few people would buy it in book format (Postman 1986).

> Being able to read a warning on a pill bottle or write about a summer vacation is not literacy and our society should not treat it so. Literacy, for example, is being

> able to fluently read and follow the 50-page argument in Paine's *Common Sense* and being able (and happy) to fluently write a critique or defense of it. (Kay 1996 p. 548)

Another example of "literacy" that Kay repeatedly mentions is the ability to hear of a disease like AIDS and to recognize that a "disastrous exponential relationship" holds:

> Many adults, especially politicians, have no sense of exponential progressions such as population growth, epidemics like AIDS, or even compound interest on their credit cards. In contrast, a 12-year-old child in a few lines of Logo [...] can easily describe and graphically simulate the interaction of any number of bodies, or create and experience first-hand the swift exponential progressions of an epidemic. Speculations about weighty matters that would ordinarily be consigned to common sense (the worst of all reasoning methods), can now be tried out with a modest amount of effort. (Kay 1994)

Surely this is far-fetched; but why does this seem so beyond our reach? Is this not precisely the point of traditional science education? We have enough trouble coping with arguments presented in print, let alone simulations and modeling. Postman's argument implicates television, but television is not a techno-deterministic anomaly within an otherwise sensible cultural milieu; rather it is a manifestation of a larger pattern. What is 'wrong' here has as much to do with our relationship with print and other media as it does with television. Kay noted that "In America, printing has failed as a carrier of important ideas for most Americans" (1995). To think of computers and new media as extensions of print media is a dangerous intellectual move to make; books, for all their obvious virtues (stability, economy, simplicity) make a real difference in the lives of only a small number of individuals, even in the Western world. Kay put it eloquently thus: "The computer really is the next great thing after the book. But as was also true with the book, most [people] are being left behind" (1995). This is a sobering thought for those who advocate public access to digital resources and lament a "digital divide" along traditional socioeconomic lines. Kay notes,

> As my wife once remarked to Vice President Al Gore, the "haves and have-nots" of the future will not be caused so much by being connected or not to the

> Internet, since most important content is already available in public libraries, free and open to all. The real haves and have-nots are those who have or have not acquired the discernment to search for and make use of high content wherever it may be found. (Kay 2000*a*, p. 395)

What is to be done, then? This sort of critique puts the education system in the United States (and most Western countries, by obvious extension) in such bad light that many are tempted to depair. Kay's project is relentless, though: with or without the school system, the attempt to reach children with powerful ideas and the means to working with them is always worthwhile. Part of the key to seeing a way through this is to remember that education does not equal school, nor does television (or any other medium) represent an essential obstacle to education. "Television," says Kay, again recalling Postman's argument, "is the greatest 'teaching machine' ever created. Unfortunately, what it is best at teaching are not the most important things that need to be learned" (1995). But in this are also the seeds of an alternative; how could different media be harnessed in such a way as to lead in a more productive direction? How can children have any "embedded cultural experience" that encourages learning logic and systems thinking? The answer isn't in the design of any particular curriculum. Rather, Maria Montessori's vision inspires Kay: putting the emphasis on children's "absorbent minds" and the freedom to play and explore.

> The objects in our system are instead a help to the child himself, he chooses what he wants for his own use, and works with it according to his own needs, tendencies and special interests. In this way, the objects become a means of growth. (Montessori 1972, p. 150)

Note that this entire conception only makes sense if we include objects—that is, artifacts, technologies, things—as bearers of practice, discourse, and culture, and ensure that we don't abstract away from the things themselves.

Vision: Necessary but not Sufficient

We have here the elements of the thinking that produced the Dynabook vision and led to its prototypes at Xerox PARC with Smalltalk and the "interim dynabook" Alto computers. The Dynabook was not, by anyone's measure, a modest project. The fascinating thing is that while Kay did not succeed in establishing a new educational model based on a new kind of systems literacy, his project did, in a different sense, succeed: Kay's sense of a future populated by millions of personal computers has indeed come true, and the accuracy with which he predicted the shape of our computing experience is uncanny. The easily portable laptop computer, with its graphic interface, connected via wireless network to a global information resource, and capable of storing and manipulating all sorts of media—as Kay described in 1972—is precisely what I am using to compose this text.[13] *But this is not the Dynabook*. For all its capabilities, the machine I am sitting in front of as I write this—and, more importantly, the set of genres governing my practices with it—rests upon a far more staid conventional understanding of literacy (and media) than Kay had in mind.

Similarly, professional programming is today heavily influenced by the object-oriented paradigm—largely defined by Kay's team. And yet, in terms of actual practice, much of it is still "a better old thing" rather than the "almost new thing" Kay had in mind. And so despite his numerous important contributions,[14] little of today's computing, personal or otherwise, comes close to the revolutionary thinking that formed the core of Kay's work, especially his sense of computing as "child's play." Contrary to Kay's method, children today are taught computing by way of systems first designed for adults. And, true to his insights, there is little that is *transformative* as a result.

It is common for fans of Kay's project to simply claim that he was "ahead of his time"—but surely this is a simplistic and superficial analysis. The question at hand, for me as an historian, is what happened? What happened to Kay's vision over the next three decades

13. In all seriousness, there is nothing prophetic about it; there are straightforward lines of influence (running largely through Apple Computer in the 1980s) leading directly from Kay's work in the 1970s to the machine I use today.

14. Kay's contributions are well recongized and indeed celebrated among computer scientists; among countless awards and distinctions, Kay received the ACM's "A.M. Turing Award" in 2004, one of the field's highest honours.

that led certain elements to take hold and indeed revolutionize the world of computing, and other elements—perhaps those most important—to remain in obscurity?

Chapter 5: Translating Smalltalk

> *The vision has been clear all along, but vision is hard to critique effectively. The various implementations we have done, on the other hand, are complete earthly artifacts, and thus admit of criticism both by ourselves and others, and this has helped to move us forward, both on the earth and in our vision.*
>
> – Dan Ingalls, 2005

Ingalls' quote speaks to an important distinction in this study: between the Dynabook and Smalltalk itself, between the *vision* and what Ingalls has called the *image*.[1] The Dynabook vision emerged powerfully and clearly in Kay's writings in the early 1970s, and he was able to coalesce a team of colleagues around him—PARC's Learning Research Group (LRG)—on the strength of that vision. But we cannot follow the trajectory of the vision itself. If we are to follow the actors, in Latour's phrase, we have to look for tangible or visible manifestations. Fortunately, in the case of the Dynabook story, the tangible and visible is provided by Smalltalk, the programming language Kay designed in 1971 and which was soon after made real by Ingalls. Smalltalk is not merely an effect of or a spin-off of the Dynabook idea; it is in many ways the embodiment of a major portion of the Dynabook—enormously conveniently so for this story. But, of course, Smalltalk itself is not the Dynabook: it is the software without the hardware, the vehicle without the driver, the language without the literature. Nevertheless, Smalltalk and its well-documented evolution provide an enormously valuable vector for the telling of the Dynabook story.

From the very beginning, there seems to have been an essential tension within Smalltalk and within its community. The tension concerns Smalltalk as the articulation of an educational vision—that is, its utopian idealism—vs. Smalltalk as a powerful innovation in computer programming and software engineering—that is, its sheer technical sweetness.[2] That being said, among the key characters in Smalltalk's history—Alan Kay, Dan Ingalls,

1. Interestingly—and almost undoubtedly coincidentally—a Smalltalk environment saves its data, state, programs, and entire memory in a file called an "image."
2. Arnold Pacey, in *The Culture of Technology*, wrote of the notion of technical sweetness, "the fact remains that research, invention, and design, like poetry and painting and other creative activities, tend to become compulsive. They take on purposes of their own, separate from economic or military goals" (1983, p. 81).

Adele Goldberg, Ted Kaehler, and a host of others—it is difficult to label anyone clearly on one side or the other of this seeming divide. While Alan Kay has remained overtly focused on the educational vision for 35 years now, there can be no denying his role as a *computer scientist*, both in Smalltalk's early design and in any number of evolutionary moves since. Adele Goldberg, hired on at Xerox PARC in the early 1970s as an educational specialist, ironically became the chief steward of Smalltalk's trajectory into industry a decade later. Even Dan Ingalls, the programmer who actually built all the major versions of Smalltalk over the years, has written perhaps more eloquently than anyone about Smalltalk's purpose to "serve the creative spirit in everyone" (Ingalls 1981). But at several key moments in the project's history, the appeal of the educational or the technical ideal has pulled it in one direction or another. With each movement, Smalltalk has been translated somewhat, into a slightly new thing.

To trace these movements is to watch the expansion of Smalltalk's 'network' in a wide variety of directions, but also to watch the translation of elements of the vision into more durable but decidedly different things. Arguably, the sheer variety of these translations and alignments—and the absence of any one clearly dominant thrust—has led to Smalltalk's marginality in any of its realms. Arguably too, this variety is what keeps it alive. I would like to note and trace here a few key *translations*, and to take the opportunity with each to point out the resulting conceptual "black boxes" that result and which go on to set the conditions for subsequent shifts. Each translation represents the ecological shifting of aspects of the project—adding new allies; allowing for new inputs and influences; conforming or reacting to constraints and threats—and each of these shifts results in a notable difference in what Smalltalk is. As Smalltalk changes, so subtly does the Dynabook vision. We will begin at Xerox PARC in the mid 1970s.

Origins: Smalltalk at PARC in the Early Years

By 1973, the Learning Research Group at Xerox PARC had an "interim Dynabook" to serve as the basis of their research efforts. The *Alto* minicomputer—arguably the first "personal

computer"—had begun to be manufactured in small quantities and distributed within Xerox PARC. Kay remembered, "It had a ~500,000 pixel (606x808) bitmap display, its microcode instruction rate was about 6 MIPS, it had a grand total of 128k, and the entire machine (exclusive of the memory) was rendered in 160 MSI chips distributed on two cards. It was beautiful" (1996, p. 534).[3] Dan Ingalls ported the Smalltalk-72 system to the Alto (they had been developing it previously on a minicomputer), thereby establishing a basic platform for the next six years' work. Kay's team originally had 15 Alto computers, and they immediately put children in front of them, though this was difficult, owing to tensions between Xerox corporate and the relatively chaotic atmosphere at PARC. Kay writes:

> I gave a paper to the National Council of Teachers of English on the Dynabook and its potential as a learning and thinking amplifier—the paper was an extensive rotogravure of "20 things to do with a Dynabook" By the time I got back from Minnesota, Stewart Brand's *Rolling Stone* article about PARC (Brand 1972) and the surrounding hacker community had hit the stands. To our enormous surprise it caused a major furor at Xerox headquarters in Stamford, Connecticut. Though it was a wonderful article that really caught the spirit of the whole culture, Xerox went berserk, forced us to wear badges (over the years many were printed on t-shirts), and severely restricted the kinds of publications that could be made. This was particularly disastrous for LRG, since we were the "lunatic fringe" (so-called by the other computer scientists), were planning to go out to the schools, and needed to share our ideas (and programs) with our colleagues such as Seymour Papert and Don Norman. (Kay 1996*a*, p. 533)

To compensate the LRG team smuggled Alto computers out of PARC (strictly against corporate regulations) and into a Palo Alto school, and also brought local kids in to work with the machines (p. 544).

3. Compare the Alto's specs with Apple Computer's first-generation Macintosh, designed a decade later. According to PARC lore, the Alto was conceived—like Smalltalk—as a result of a bet, and the bravado of its creators. With money diverted from the LRG budget, Chuck Thacker from the PARC's Computer-Science Lab intiated the project while the executive in charge of the lab was away, having boasted that they could create a whole machine in three months (Kay 1996, p. 532).

Figure 5.1: Kids in front of Alto computer (from Goldberg 1988)

Adele Goldberg writes:

> Most of the educational experimentation was done with specially conducted classes of students ages 12–13. These classes were held in cooperation with a local high school's mentally gifted minors program. The students were driven to PARC during the school day. Saturday classes were held for the children of PARC employees. (Goldberg 1998, p. 62)

Smalltalk-72 running on the Alto machines proved good enough for the first round of research. Kay and LRG colleague Diana Merry first worked on implementing an overlapping-window mouse-driven screen interface, with text in proportional fonts. LRG team member Steve Purcell implemented the first animation system, and Ted Kaehler built a version of turtle graphics for Smalltalk. Larry Tesler created the first WYSIWYG page-layout programs. Music synthesis had already been implemented before the Alto, and so this was moved over and substantially developed on this first generation platform.

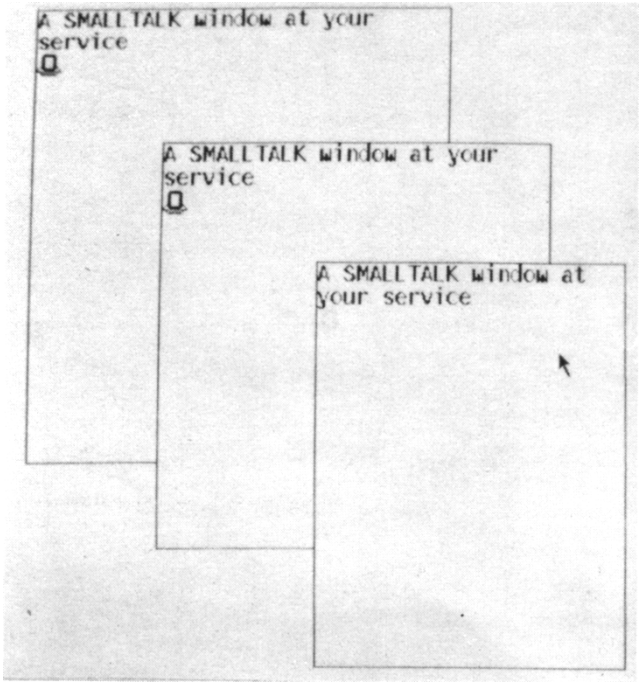


Figure 5.2: Original overlapping-window interfaces (from Kay & Goldberg 1976, p. 16).

All of this work was considerably enhanced when Ingalls, along with Dave Robson, Steve Weyer, and Diana Merry, re-implemented Smalltalk with various architectural improvements (a version unofficially referred to as *Smalltalk-74*), which brought enormous speed improvements to the system (Kay 1996*a*, pp. 542–543; Ted Kaehler, personal communication, Nov 2005).

With the Smalltalk-72 system, Adele Goldberg worked substantially on a scheme merging turtle graphics and the new object-oriented style, using the simple idea of an animated box on screen (named "Joe"). The box could be treated like a Logo turtle—that is, given procedural commands to move around the screen, grow and shrink, and so on; but it could also act as a 'class' from which specialized kinds of boxes could be derived.

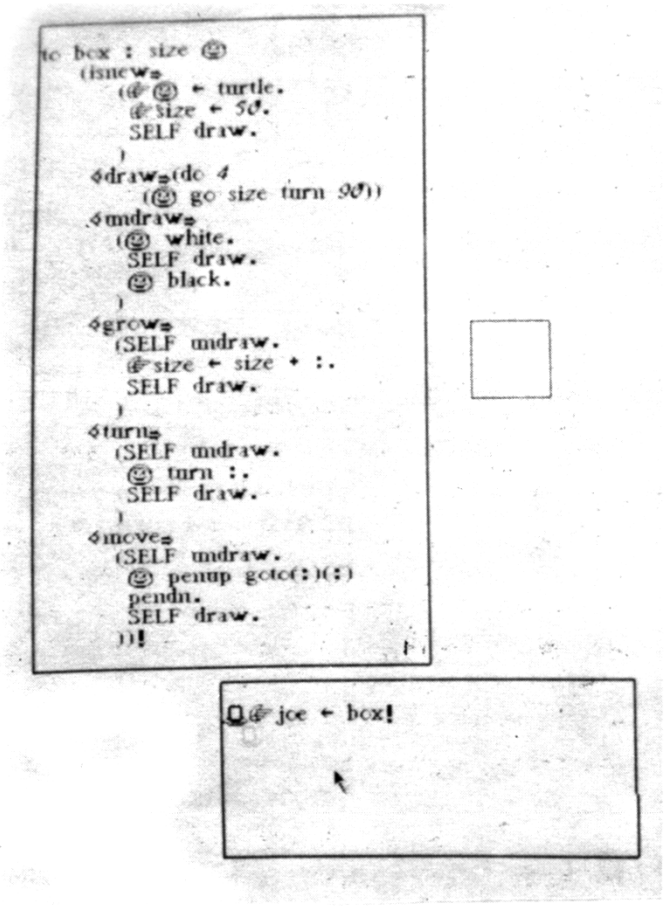


Figure 5.3: Adele Goldberg's Joe Box in action (Kay & Goldberg1976, p. 47)

Kay later reflected,

> What was so wonderful about this idea were the myriad of children's projects that could spring off the humble boxes. And some of the earliest were tools! This was when we got really excited. For example, Marion Goldeen's (12 yrs old) painting system was a full-fledged tool. A few years later, so was Susan Hamet's (12 yrs old) OOP illustration system (with a design that was like the MacDraw to come). Two more were Bruce Horn's (15 yrs old) music score capture system and Steve Putz's (15 yrs old) circuit design system. (Kay 1996, p. 544)

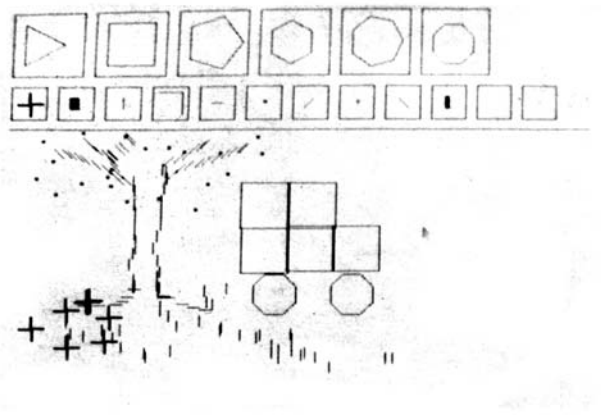


Figure 5.4: Marion's painting system (from Kay & Goldberg 1976, p. 33)

As exciting as these early successes must have been (and frankly, as impressive as they still sound today), the limitations of this early work—appearing along two fundamentally different axes—would significantly shape further development and point it in divergent directions.

Educational limitations

A serious problem Kay's team encountered was the extent to which children and novice users hit a "wall" or critical threshold in the complexity of their designs and constructions.[4] Kay reflects:

> The successes were real, but they weren't as general as we thought. They wouldn't extend into the future as strongly as we hoped. The children were chosen from the Palo Alto schools (hardly an average background) and we tended to be much more excited about the successes than the difficulties. ... We could definitely see that learning the mechanics of

Xerox PARC in the Press

An article in *Rolling Stone* magazine (Dec 1972) by Stewart Brand entitled "Spacewar! Fanatic Life and Symbolic Death Among the Computer Bums" documented the common ground between the (post-) hippie counterculture and California-based computer researchers. In Brand's exposé of Xerox PARC, Kay and his colleagues are portrayed as long-haired *freaks* in the Bay Area style. The article, which features a good number of Kay's enthusiastic pronouncements, centers on the hacker tradition of *Spacewar!*, arguably the world's first video game, which ironically embodies the interactive, networked goals of the ARPA project—*pace* Paul Edwards' sinister "closed world" trope (1996).

The article notes, "Though no one has done it yet, Alan Kay is convinced a modest *Spacewar!* could be built cheap," followed by some quick thoughts from Kay on cobbling together the electronics for a *Spacewar!*-capable machine ("for 700 bucks or something like that"). Provided is a short program listing for *Spacewar!* by Kay—in Smalltalk-72.

With its proclamation, "Ready or not, computers are coming to the people," the Brand article is one of the most revealing historical documents on the whole ARPA-PARC endeavour; Xerox responded with a five-year publication ban on the researchers at PARC.

> the system was not a major problem. The children could get most of it themsleves by swarming over the Altos with Adele's JOE book. The problem seemed more to be that of design. (Kay 1996*a*, p. 544)

Kay agonized over the difficulty of teaching a group of "nonprogrammer adults" from PARC to construct a simple rolodex-like information manager application in Smalltalk. Kay noted how they moved more quickly than the children, but this critical threshold still appeared earlier than he had anticipated:

> They couldn't even come close to programming it. I was very surprised because I "knew" that such a project was well below the mythical "two pages" for end-users we were working within. Later, I sat in the room pondering the board from my talk. Finally, I counted the number of nonobvious ideas in this little program. They came to 17. And some of them were like the concept of the arch in building design: very hard to discover, if you don't already know them.
>
> The connection to literacy was painfully clear. It isn't enough to just learn to read and write. There is also a literature that renders ideas. Language is used to read and write about them, but at some point the organization of ideas starts to dominate mere language abilities. And it helps greatly to have some powerful ideas under one's belt to better acquire more powerful ideas. (p. 545)

Despite the intractibility of this problem, even three-and-a-half decades later, Kay puts the focus in the proper place: the issue is a cultural one, rather that a technical problem that can be fixed in the next version. This is an issue that still hasn't been significantly addressed in educational technology community, despite any number of attempts to create computing environments for children or 'novices.' Goldberg's emphasis on *design*, as in the "Joe box" example, seemed to Kay to be the right approach. But it was clear that the specifics of just how to approach the issue of design eluded them, and that they had a very long way to go.

Technological limitations

As innovative and successful as Smalltalk-72 had proved, its weaknesses soon showed through: weaknesses inherent in Kay's original parsimonious design. In Smalltalk-72, the

4. Ted Kaehler more dramatically called it "the cliff" (Kaehler, personal communication, July 7, 2004).

actual syntax of the language in any particular instance was defined by the code methods attached to the particular object receiving the message. "This design came out of our assumption that the system user should have total flexibility in making the system be, or appear to be, *anything* that the user might choose" (Goldberg & Ross 1981, p. 348). The trouble with this approach is that the flexibility of the system tends to preclude consistency. The team agreed that the flexibility of Smalltalk-72 was beyond what was desirable (Kay 1996*a*, p. 547). Adele Goldberg and Joan Ross explained:

> Our experience in teaching Smalltalk-72 convinced us that overly flexible syntax was not only unneccesary, but a problem. In general, communication in classroom interaction breaks down when the students type expressions not easily readable by other students or teachers. By this we mean that if the participants in a classroom cannot read each other's code, then they cannot easily talk about it. (Goldberg & Ross 1981, p. 348)

A new Smalltalk seemed to be the next step. But the team were not in total agreement about how exactly this should be accomplished.

Smalltalk's Initial Transformation at Xerox PARC

Translation #1: A Personal Computer for Children of All Ages becomes Smalltalk-80

In early 1976, Alan Kay worried that his project was getting off track, and concerns with the design and implementation of the system were leading the LRG farther away from research with kids. He wanted to refocus, and so he took his team on a three-day retreat under the title "Let's Burn Our Disk Packs"—in other words, let's scrap what we have now, return to first principles, and begin again (Kay 1996*a*, p. 549). He explained his feeling with reference to Marshall McLuhan's chestnut, "man shapes his tools, but thereafter his tools shape him," and wrote, "Strong paradigms like Lisp and Smalltalk are so compelling that they *eat their young*: when you look at an application in either of these two systems, they resemble the systems themselves, not a new idea" (p. 549).

Not surprisingly, the people who had spent the past four years building Smalltalk were not keen on throwing it all away and starting from scratch. Dan Ingalls, especially, felt that a new Smalltalk was indeed the right direction, and by now he had some well-developed ideas about how to do it better. Ingalls thus began the design of a major new version, called *Smalltalk-76*. This proved to be a turning point, as Ingalls' new thrust with Smalltalk would generate enormous momentum, cementing the technical foundations of a whole paradigm of computer programming.

A response to the technical limitations the LRG had found in their work with Smalltalk-72, Smalltalk-76 clearly identified and established itself as the paradigm for object-oriented programming. The Smalltalk-76 language and environment was based on a cleaner and more consistent architecture than in Smalltalk-72: here, everything in the system was an *object*; objects communicate by passing *messages*; objects respond to messages sent to them via the code in *methods*. Furthermore, every object is an instance of a *class*; "the class holds the detailed representation of its instances, the messages to which they can respond, and methods for computing the appropriate responses" (Ingalls 1978, p. 9). These classes are arranged in a single, uniform hierarchy of greater and greater specialization. Much of the actual practice of programming in such a system is the definition of new (lower) levels of this hierarchy: "subclassing," that is, taking a class which provides some functionality and extending it by defining a new class which inherits the old functionality plus some specialization. Goldberg and Ross wrote:

> The Smalltalk-76 system was created primarily as a basis for implementing and studying various user-interface concepts. It gave the users, mostly adult researchers, further ability in refining existing classes through the use of subclassing. This meant that the programmer could now modify a running model without creating a change to already existing examples of that model. *Programming-by-refinement*, then, became a key idea in our ability to motivate our users. (Goldberg & Ross 1981, p. 354)

Ingalls and his colleagues made the system into more and more of a working environment for themselves; more and more of the development of the system was done within the

Smalltalk-76 system itself. A key point which is often overlooked in glosses of Smalltalk's capabilities is that the system was completely live or "reactive" (Ingalls 1978), and so such changes to the system could be made on-the-fly, in real time (imagine changing how Microsoft Word works in the middle of writing a paragraph). In fact, parts of current Smalltalk environments were developed in the Smalltalk-76 environment at Xerox PARC.[5]

There emerged a unique community of practice within Xerox PARC surrounding the Smalltalk-76 environment. Even Kay was "bowled over in spite of my wanting to start over. It was fast, lively, could handle big problems, and was great fun." The momentum of this community of practice firmly established some of Smalltalk's enduring and influential features: a multi-paned class "browser" which allowed one to quickly and easily traverse all the classes and methods in the system; an integrated, window-based debugging environment, in which errors in the system pointed the user directly to the methods requiring fixes; and the first consistent, system-wide windowing user interface, the prototype for the ones we all use today.

5. The significance of this is easy to miss, but the claim about lineage is not a trivial thing. Nearly all programming environments distinguish between a program—normally treated as a static text—and its execution. Smalltalk, like Lisp before it, can be written while the program is running, which means that software can be modified from within. Since Smalltalk is a live computing environment in which code dynamically co-exists with transient objects like state information and user input, later versions of Smalltalk were created *within* a Smalltalk-76 environment, and more recent versions constructed *within* these. In a sense, the Smalltalk-76 environment has upgraded *itself* a number of times in the past thirty years. See Ingalls 1983, pp. 24–25.

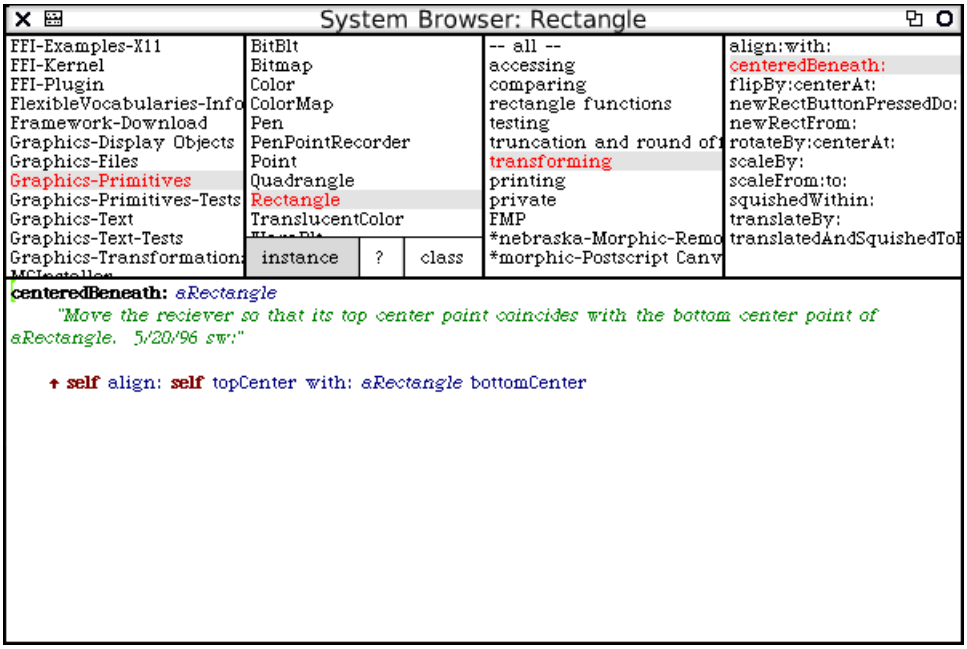


Figure 5.5: A Smalltalk "browser," showing classes and methods arranged in arbitrary categories. This browser is a direct descendant of the ones developed by Larry Tesler in Smalltalk-76.

By the late 1970s, Smalltalk's *generic* qualities had begun to be evident: the consistency of the system and the design methodologies it encouraged had an effect on the practices of its users. I deliberately mean to treat Smalltalk as an actor in this sense; here is an example of a tool which, once shaped, turns and shapes its users—or "eats its young," as Kay put it. This is the system which excited computer scientists about object-oriented programming, and the user-interface genre established by Smalltalk-76 was adhered to in Apple and Microsoft's later systems. A broad set of practices and approaches coalesced around Ingalls' new design:

> When programmers started writing class definitions in these browsers, a new era of design began. The average size of a method tended to correspond to the screen space available for typing the method text (which consisted of around seven message expressions in addition to the method header information). Software evolved (actually it felt like software was molded like clay). The programmer could write a partial class description, create an instance to try out its partial capabilities, add more messages or modify existing methods, and try these changes out on that same instance. The programmer could change

the behavior of software describing an instance while that instance continued to exist. (Goldberg 1998, p. 64)

The educational research that followed on the development of Smalltalk-76, however, was not aimed at children, but at adult end-user programming. In 1978, Adele Goldberg led the LRG team through a watershed experience: they brought in Xerox upper management and taught them to construct a business simulation using Smalltalk. A key departure here was that these end-users were not exposed to the entire Smalltalk-76 environment, but to a special simulation framework designed for the exercise (Kay 1996*a*, p. 556ff; Goldberg 1998, pp. 65–66). Much of the research in the late 1970s seems to have taken the form of such domain-specific environments and focusing on *design*, rather than teaching the programming environment itself (Goldberg 1979; Goldberg & Ross, 1981).

Parallel to the Smalltalk-76 development was Kay's work on a machine called the *NoteTaker*. This was the first "portable" computer in the sense in which we understand it today; it was a lot bigger than a laptop, but Kay writes that he did use it on an airplane (1996*a*, p. 559). This isn't just a footnote; the important point about the NoteTaker, from the standpoint of Smalltalk's trajectory is that for the first time since the Alto's introduction, Smalltalk was made to run on a non-Xerox processor. Ingalls and colleague Bruce Horn ported the Smalltalk-76

What's Different about Smalltalk

A few key features distinguish Smalltalk from almost all other software development environments.

In Smalltalk, "everything is an object" capable of sending and receiving messages; there are no other conceptual constructs. Other object-oriented languages, such as C++ and Java use objects along with other constructs like functions or specific data types drawn from mainstream programming traditions. Smalltalk's architecture and syntax are very simple, since everything in the system behaves in the same way.

Smalltalk is not a "compiled" language (like Pascal, C, and C++), in which source code files are batch-translated into machine-readable executables. Nor is it an "interpreted" language (like BASIC, Perl, Python, in which source code files are "run" by an platform-specific interpreter. Instead, Smalltalk (like Java) runs in a "virtual machine," "bit-identically" on a wide variety of platforms. Java's virtual-machine architecture was drawn from Smalltalk.

Smalltalk is also its own development and runtime environment. Rather than executing programs on top of an underlying operating system layer which maintains file input-output and a filesystem, Smalltalk runs in an "image"—a single, live "file" that manages its own memory use, reads and writes itself to disk transparently when required, and which permanently maintains the entire state of the environment. It is this 'live' and persistent image which allows Smalltalk to be changeable "on the fly"—other languages require that one make changes to source code files and then recompile or re-run the files in order to make a change. Smalltalk images from the days of Smalltalk-80 (and even Smalltalk-76) are still in use, having been bootstrapped into modern versions.

system over to the NoteTaker (making *Smalltalk-78*), which was built with the new, inexpensive microprocessor chips (such as would appear in the first microcomputers). So, despite the NoteTaker project being officially cancelled by Xerox management in 1978, Smalltalk had taken its first steps toward portability—that is, independence from Xerox' own hardware. Goldberg reports that further work toward making Smalltalk run on other vendors' hardware was key to its continued evolution (1998, p. 69ff), convincing the team that

> ...Smalltalk would run well on standard microprocessors. We no longer needed to rely on microcoding Xerox' proprietary machines, so we decided it was time to expand the audience for Smalltalk. We decided to create a Smalltalk that the rest of the world could use. [...] In 1979 we asked Xerox for the right to publish Smalltalk, the language and its implementation and the applications we had built to test the Smalltalk model of computing. Xerox officially gave this permission, remarking that no one inside Xerox wanted Smalltalk. (Goldberg 1998, p. 71)

Outside Xerox, there *was* interest, and the popular history of computing records the occasion of Apple Computer's Steve Jobs and his team visiting Xerox in 1979, and coming away with substantial inspiration for their Macintosh project. Interestingly, what Jobs and his team really took away from their 1979 visit was the *look* of what Kay's team had designed and not so much of how it worked; Jobs was so bowled over by the windows-and-menus interface that he ignored the dynamic object-oriented development environment and the local-area network connecting the Xerox workstations.

The symbolic importance of this event relates to these ideas 'escaping' from Xerox. The distillation of the Smalltalk-76 environment over the following three years into *Smalltalk-80* made this motif central. In preparing Smalltalk-80 for release, what was required was to abstract the language from any hardware assumptions in order to allow implementations on any number of target platforms (Goldberg 1998, p 73).

Significantly, Kay was not a part of this development. In 1979, Kay took a sabbatical from Xerox, and did not return. Adele Goldberg led the group toward the definition of Smalltalk-80 and the publication of a series of books (Goldberg & Robson 1983; Krasner 1983; Goldberg 1984) and a special issue of *BYTE* magazine (Aug 1981) with a cover illustration showing a colourful hot-air balloon ascending from a tiny island with an ivory tower. But Smalltalk was escaping to where? Certainly not to schools and schoolchildren; rather, Smalltalk-80 was headed for professional systems programming—electronics, banking, shipping—and academic computer science research. The flexibility and elegance of Smalltalk's development environment won it a small but dedicated following of systems programmers; this would be what Smalltalk was known for in programming circles. The resulting "black box" (in Latour's sense) was Smalltalk as an interesting dynamic programming environment for research and systems modelling, but far from the mainstream of either professional software development or personal computing.

Xerox PARC in the Press – II

After Stewart Brand's 1972 article appeared, there were no publications from anyone in Kay's team until 1977, when Kay and Goldberg's paper, "Personal Dynamic Media," and Kay's *Scientific American* article, "Microelectronics and the Personal Computer," finally revealed a glimpse of their research to the outside world. But there would be no real details about Smalltalk until the 1980s. A *BYTE* Magazine cover from 1978 made oblique reference to the "magical kingdom of Smalltalk," isolated on a rugged island, an image that would reappear on a 1981 cover, now showing Smalltalk escaping from the island via hot-air balloon (an image inspired by Jules Verne's *The Mysterious Island*, according to Dan Ingalls).

Translation #2: From educational research platform to software development tool

Xerox licensed Smalltalk in 1980 to four hardware companies who had their own software divisions and could therefore participate in the documentation of its implementation in different contexts: Hewlett Packard (HP), DEC, Apple, and Tektronix. Of these, electronic instrument manufacturer Tektronix (an electronic equipment manufacturer rather than a computer company *per se*) did the most with Smalltalk, offering it with a short-lived line of research workstations, and also embedding in the hardware of its popular line of oscilliscopes (Thomas n.d.). Significantly, Smalltalk got a bigger boost in the late 1980s with the formation of a spinoff from Xerox called ParcPlace Systems, in which Goldberg and

colleagues commercialized Smalltalk, selling licenses to more companies and maintaining a portable base system which would hedge against the language's fate being tied to any one hardware platform (Goldberg 1998, p. 80ff). Ultimately, two Smalltalk licensees came to dominate: IBM and Digitalk—the latter was a spinoff from Italian business products company Olivetti, which ultimately merged with Goldberg's ParcPlace Systems; the company was acquired in 1999 by Cincom, a major software consulting house.

If this historical detail sounds somewhat arcane and far removed from the trajectory of the Dynabook vision, it should. This later history of Smalltalk, from the early 1980s on, has a decidedly different character from that which preceded it. The focus had shifted entirely away from education (with a handful of minor exceptions[6]) and toward leading-edge computer science research and development. Much of the activity in the Smalltalk community was academic, centered around teams at the Xerox PARC of the 1980s as well as research at Universities of Massachussetts, Washington, Carleton, Tokyo, Dortmund, and others worldwide (ibid.). Ironically, far from its origins in personal computing, Smalltalk in use is found in the realm of big systems development: banking and finance, importing/exporting and shipping, health care, insurance, and so on.[7] The website for Cincom Smalltalk boasts, "how the French fries you get at McDonalds are sorted by Cincom Smalltalk."

Smalltalk's greatest impact on the computing world, however, was its role in the establishment of *object-oriented* programming and design, which by now has become one of the major genres of contemporary information technology. Smalltalk may have been the technology at the core of this movement in the 1980s, but it was quickly overtaken by much larger populations of developers working in the C++ language (which, strictly speaking was derived from the earlier *Simula* language rather than Smalltalk, and which added object and class constructs to the popular C programming language). C++ was a much smaller concep-

6. Adele Goldberg and Joan Ross wrote an article in the 1981 *BYTE* special issue on Smalltalk entitled "Is the Smalltalk-80 System for Children?"—the answer was a qualified yes, but it would appear that this article serves mostly to establish Smalltalk-80's intellectual tradition rather than to introduce new material. Smalltalk found favour as a teaching language in a few academic computer science departments, but remained very far from the mainstream.

7. See http://www.whysmalltalk.com

tual and practical leap for mainstream programmers used to working in static, procedural languages like C or Pascal; though the consequence of that 'shorter leap' also means that C++ has been called the worst of both worlds. Despite this, C++ grew in the 1990s to be the dominant object-oriented language, and its popularity was such that object-oriented programming became the new mainstream. Consider, as a measure of this, the fact that the U.S. "Advanced Placement" curriculum in computer science shifted to C++ from Pascal in 1999.

In 1996, Sun Microsystems released the *Java* language and development platform, an attempt to re-invent software development with the Internet in mind. Java is an object-oriented language much closer in spirit to Smalltalk, at least in that it was designed from the ground up with objects in mind (unlike C++, which was an adaptation of an older language and conceptual model), and with a virtual-machine architecture like Smalltalk's to ensure portability across a wide variety of platforms. According to programmer mythology,[8] Sun Microsystems wanted "an industrial strength Smalltalk, written in C++." They got neither, but what Java did represent after its late-'90s release was an enormous shift of programming practice—and, perhaps more importantly, discourse—away from C++ (The US Advanced Placement curriculum abandoned C++ for Java in 2003). Sun Microsystems spent the better part of the next decade fighting with Microsoft over this shift and who would be in control of it. Meanwhile, the Smalltalk community—consultants and developers at IBM, Digitalk, and a few others—continued on in the shadows of these enormous efforts. It is worth noting that while millions of people work in C++ and Java and Microsoft's related .NET on a daily basis, the almost ubiquitous characterization of these environments is that they are overly complex, badly designed and implemented, and a general curse on their users. The way Smalltalk developers talk of their chosen environment couldn't be more different.

8. My source for this 'mythology' is the collected wisdom and commentary on contemporary programming practice at the original WikiWikiWeb (http://c2.com/cgi/wiki). The WikiWikiWeb was begun in the mid 1990s by Ward Cunningham, a Smalltalk programmer at Tektronix who wanted to host a collaboratively authored and maintained collection of software "design patterns"—a methodology inspired by architect Christopher Alexander's *A Pattern Language* (1977). Cunningham and various colleagues (none of whom are still at Tektronix) became key figures in the OOP community, associated with the "design patterns" movement (see Gamma et al. 1995), and are also key figures in the newer "eXtreme Programming" movement (see, eg. Beck 2000). The WikiWikiWeb remains a central repository of commentary on these topics, boasting over 30,000 'pages' of collected information.

The second major translation of Smalltalk, then, is from a research project—at its origin an *educational* research project—to its marginal place within a much larger current of industrial practice in object-oriented programming. Smalltalk's status within this larger current sometimes reads like an origin myth ("*In the beginning, there was Smalltalk...*"). The related black-boxing of Smalltalk in the context of this historical shift relegates it to an intellectually interesting but ultimately 'academic' system, too far from evolving mainstream concerns to make much practical difference. Far from being the revolutionary step Kay had hoped, Smalltalk was merely subsumed within the emerging object-oriented paradigm.

Translation #3: From "designers" to "end-users"

It is against these large-scale, corporate systems trends that the Dynabook's trajectory through the 1980s and 1990s must be evaluated. After 1980, Smalltalk almost completely shed its educational connections; very little of Smalltalk-80 was ever seen by children. Ironically, it was Adele Goldberg, who came to Xerox PARC as an educational specialist and who led the research with children there for years, who now led Smalltalk's move into the wider world of professional programming.[9] It is important to reflect on just how far Smalltalk had travelled from the Dynabook vision, and it would have been a reasonable observation in the mid 1980s that the two ideas had finally parted. Benedict Dugan's commentary on this shift invokes Frankfurt-school theories of "technical rationalization":

> Clearly, at some point, the original, idealistic goals of Kay and company became commercialized. By commercialized, I mean that the design focus shifted away from social and political concerns, to an interest in efficiency. By exploiting the ability of class hierarchies to organize knowledge and share code, the designers created a language which was promoted for its ability to facilitate extremely efficient software engineering. Lost was the powerful notion of a programming system which would amplify the human reach and make it possible for novices to express their creative spirit through the medium of the computer. (Dugan 1994).

9. In 1984, Adele Goldberg became president of the Association for Computing Machinery, computing's largest professional association. Goldberg did remain connected to education, however; her work in the 1990s with NeoMetron employed Smalltalk in the design of learning management systems (Goldberg et al. 1997).

Despite the finality of Dugan's statement, Smalltalk was far from finished; the contributions to computer science embodied in Smalltalk are still being realized and reconsidered today. I will not go into detail here on the long-term impact of Smalltalk on software engineering. Instead, I want to focus specifically on the rise of user-interface development.

Conventionally, the graphical user-interface as we know it today descends more or less from the Smalltalk environments at Xerox in the 1970s, via Steve Jobs' visit to Xerox PARC and subsequent design of Apple's Macintosh computers; what Xerox couldn't bring to market, Apple could, and in a big way. The actual story is a little more complicated than this, not surprisingly.

In the first place, Xerox did attempt to commercialize some of the personal computing research that came out of PARC. In the late 1970s, a machine called the Xerox *Star* was developed and it was sold in the early 1980s.[10] The Star was an attempt to market what Kay calls the "PARC genre" of computing: a pointer-driven graphical user interface, rich document-production tools ("desktop publishing"), peer-to-peer networking, and shared resources like a laser printer. The Star's failure commercially has probably more to do with its selling price—close to $20,000 each, and they were sold in clusters of three along with a laser printer—in an era when the "personal computer" was being defined by Apple and IBM's machines costing around $3000. Nevertheless, the Star is the machine which first brought the modern graphical desktop environment to market; while overlapping windows and menu-driven interaction were pioneered in Smalltalk, the process of turning these ideas into a packaged product and sold to an audience happened with the development of the Star. Interestingly, the use of *icons*—not a feature of the Smalltalk interface—was pioneered in the Star interface as a means of giving a direct-manipulation interface to mundane, hardware-defined things like disks and printers and files.

I do not mean to give the impression that the Xerox Star was completely distinct from the Smalltalk project—there was some significant overlap in the personnel of the two

10. The Xerox Star was sold in reasonably large quantity for the day; the Wikipedia entry on the Star states that about 25,000 of the machines made it to market—as corporate office technology. http://en.wikipedia.org/wiki/Xerox_Star (RetrievedSept 15, 2005).

projects—but rather to point out yet another considerable translation which occured in the packaging and marketing of the Star. In Kay's Dynabook concept, end-users were seen as *designers* and *developers*; system tools were accessible from top to bottom, and the "late binding" philosophy led to an expectation that the details of how a user actually worked with a system would be defined *ongoingly* by that user.[11] This clearly presents difficulties from a business perspective; how on earth does one market such a concept to a potential audience? It is far too vague. The Xerox Star, then, was a distillation of one possible scenario of how typical office-based end-users would work. The Star operating system was not Smalltalk-based, so it would not be possible to easily change the configuration; instead, the Star's developers worked according to a now-commonplace model: usability research. They were able to draw upon several years of internal Xerox use of the Alto computers, with and without Smalltalk—there were over 2000 of them in use at Xerox in the 1970s—and they developed a detailed model of tasks and use cases: what would end-users want to do with the Star, how would they go about it, how should the user interface be structured to enable this?

The shift here is from a notion of participatory designers (in Kay's conception) to *end-users* as we understand the term now. For Thierry Bardini and August Horvath, in their article, "The Social Construction of the Personal Computer User" (1995), this is the point where the end-user role is formally established. Employing the language of actor-network theory, they write,

> The first wave of researchers from SRI to PARC helped in opening the concept of design by the reflexive user, and the second wave got rid of the reflexive user to create a methodology of interface design based on a user model and task analysis. In this last translation, the very utility of the reflexive user ... was questioned.
>
> The result was a new set of principles for the design of the user interface and its new look and feel: icons and menus. The first step of this new methodology is also the last that we consider for this part of the history. Here begins the

11. Note that in this formulation, "user" refers to an actual individual, as opposed to an hypothetical "User" for whom the system has been designed.

> negotiation with *real users* over the script of personal computing. (Bardini & Horvath 1995 [italics added])

There are two substantial black boxes emerging here: first is Bardini & Horvath's notion of "real users;" second, and at least as influential, is the trope of "user friendliness," which is the target of user-centered design and possibly the key selling point of the microcomputer revolution, especially since the Macintosh.

But this is all too *pat*, both as history and as pedagogy. Bardini and Horvath seem content to close the box and write the history as done at this point—what follows is the unfortunate popular history of personal computing with its monolithic end-users and engineers. I am not prepared to close the box here, and neither are the community of people surrounding Kay and the Dynabook idea, as we shall see. In my reading of this history, it is imperative that we question whether the "realization" (in Bardini and Horvath's language) of the *User* and the attendant reification of the qualities of user-friendliness (ease of use, ease of learning, not demanding too much of the user) is something which we are prepared to accept. It seems to me that the "reflexive users" of the early PARC research are in fact *more real* than the hypothetical one(s) inscribed in User-Centered Design—which performs a substitution not unlike what *focus groups* do for markets or audiences. The earlier reflexive users at least had agency in their scenarios, as they actively shaped their computing media. The later Users, though in vastly greater numbers, must be satisified with being folded into a pre-shaped role. It is, of course, indisputable that this latter version has become the dominant one. But one of the consequences of this closure is the rise of a genre of computing literature (especially within education) which diagnoses the problems stemming from the cultural disconnect between "engineers" and "end users" (e.g., see Shields 1995; Rose 2003). This diagnostic tendency is rarely constructive (see Papert's 1987 defense of computer cultures); rather, it more effectively serves to further reify these oppositional roles.

It is significant, I think, that Kay's original (and ongoing) conception of personal computing (especially in education) is an alternative to the now-commonplace notion of

end-users. Bardini and Horvath suggest that Kay's "reflexive users" are an artifact of history—back when computers were only for 'computer people'—but I am not yet/quite convinced. Nor are the proponents of a movement in technology design and policy which emerged in Scandinavia in the late 1970s and early 1980s called *participatory design* (Ehn 1988), which saw a specifically political dimension in the division of labour—and power—between workers (increasingly seen as end-users) and the designers and engineers representing the interests of corporate power. The participatory design movement sought to address this head-on, with direct involvement from labour unions.[12] It is, I think, instructive that a movement which significantly addresses issues of power in computing environments should seek to re-inscribe the user.

The Microcomputer Revolution of the Late 1970s

The history of the advent of the "PC"—the personal microcomputer[13] as we have come to know it—has been copiously documented and is not a topic I will devote much time to here; there are several standard histories, the PBS documentary series *Triumph of the Nerds* (Cringely 1996) is probably sufficient as a touchstone. The storyline has become mostly conventional: unkempt hackers in their (parents') northern California garages discovered what IBM—the market leader in computing—had missed, and, as a result, tiny startup companies like Apple and Microsoft had the opportunity to make hay for themselves, eventually eclipsing IBM. Significantly, these companies—aided in no small part by IBM itself—succeeded in making the personal computer a necessary part of modern life: we soon became convinced that we needed them in every office, every home, every classroom.

12. Interestingly, one of the leading figures in the early 1970s Scandinavian "experiment" was Kristin Nygaard, who had co-designed the original object-oriented *Simula* programming language a decade before.

13. I will use the term "microcomputer" here to distinguish between Alan Kay's conceptualization of a "personal" computer and the small, inexpensive hobby- and home-targeted microcomputers which emerged in the late 1970s and early 1980s. The latter came to be known as "personal computers" especially after IBM branded their microcomputer offering as such in 1981.

Translation #4: From a software research tradition to a "gadget" focus

The interesting thing about the conventional story of the microcomputer vanguard is that what had been accomplished at Xerox PARC in the 1970s is almost entirely absent; their creators (that is, proto-billionaires like Steve Jobs and Bill Gates) operated in a world nearly perfectly isolated from the kind of thinking Alan Kay was engaging in. Apple Computer's promethean role is mostly as a hardware manufacturer: they created devices—boxes—that a computer hobbyist could afford. Microsoft's part was to market a rudimentary operating system for IBM's entry into the PC market. Both of these contributions—significant as they were in hindsight—were astonishingly unsophisticated by PARC's standards. Kay reportedly "hated" the new microcomputers that were coming out: "there was no hint that anyone who had ever designed software was involved" (1996*a*, p. 554).

But this is not simply a matter of economics: the difference is not explained by the size of the budgets that separated, for instance, Xerox from Apple in 1978 or 1979. It is rather a cultural difference; Kay's work—and that of his colleagues at PARC—drew upon a lengthy academic tradition of computing: these were all people with PhDs (and not necessarily in computer science, as Kay points out, but in "established" disciplines like mathematics, physics, engineering, and so forth). Apple founders Jobs and Wozniak were hobbyists with soldering irons, more in the tradition of hot rodding than systems engineering. Bill Gates was a Harvard dropout, a self-taught programmer who saw the business potential in the new microcomputers.

Not that these self-styled pioneers were positioned to draw on PARC's research; Xerox publications were few and far between, and while the Dynabook and Smalltalk work was not secretive, what was published broadly was not the sort of thing that a self-taught, garage-based hacker could work with, despite Kay's best intentions. Even today, with the accumulated layers of three decades of computing history at our fingertips, much of the PARC research comes across as slightly obscure, much of it is still very marginal to mainstream computing traditions. The microcomputer revolution was primarily about hardware, and there is no doubt that much of its early energy was based in a kind of gadget fetishism. As

this early enthusiasm matured into a market, the resulting conceptual black box was the PC as a thing on your desk, a commodity. Who could conceive of how software might become a marketable item? It must have seemed more of a necessary evil than something important in and of itself, at least until the hardware market was sufficiently established for tools like *VisiCalc*—the first "killer app"—to be appreciated.

Translation #5: From a research focus to a market focus

The pioneers of the microcomputer succeeded most importantly as marketers, turning the hobbyist's toy into something that 'everybody' needed. In this respect their place in history is secured. The scaling-up of the world of computing from what it looked like in 1970—according to PARC lore, of the 100 best computer scientists in the world, 80 of them were working at PARC—to its size even a decade later, is difficult to encapsulate, and I won't try. Suffice it to say that it complicated Kay's vision enormously. But wouldn't, one might argue, the appearance of a personal computer on every desk be right in line with what Kay was driving at? Here, seemingly, was the prophecy fulfilled; what's more, right from the beginning of the microcomputer age, advocates from both sides of the table—at schools and at technology companies—were trying to get them in front of kids.

Necessary, perhaps, but not sufficient. To look at it a little more closely, the microcomputer revolution of the late '70s and early '80s represents more of a cusp than a progression. As for the details—frankly, the early microcomputers were pretty useless: their hobbyist/tinkerer heritage made them more like gadgets than the personal media tools Kay had envisaged. Market pressures kept them woefully underpowered,[14] and the lack of continuity with the academic tradition meant the software for the early microcomputers was *uninspired*,[15] to say the least. The process of turning the microcomputer into an essential part of modern life was a much bumpier and more drawn-out process than the popular mythology

14. Although Kay's team in the 1970s foresaw a $500 personal computer, what they were actually working with cost vastly more; the transition to mass-produced (and therefore inexpensive) machines was not well thought out at Xerox. What was possible to create and bring to market for a few thousand dollars in the early 1980s was still a far cry from the Altos.

15. This reference to inspiration refers to software traditions beyond PARC too; it was not until the early 1990s (and publicly-accessible Internet) before Unix-based operating systems—another software tradition with roots in the 1970s—made any real impact on the PC market.

suggests. The question, "what are these things good for?" was not convincingly answered for a good many years. Yet the hype and promise dealt by the early advocates was enough to drive things forward. Eventually, enough black boxes were closed, the answers were repeated often enough to begin to seem right ("yes, I *do* need Microsoft Word"), and the new application genres (spreadsheets—desktop publishing—video games—multimedia—etc.) were layered thickly enough that it all began to seem quite 'natural.' By the time the Internet broke through to public consciousness in the early 1990s, the personal computer was all but completely established as an indispensable part of daily life, and the rhetoric of determinism solidly won out: you really can't function without one of these things; you really will be left behind without one; your children will be disadvantaged unless you get on board.

The resulting black box from this translation was the identification of the computer and computer industry as the "engine of the economy," with the various elements of computing firmly established as market commodities. But what had happened to the Dynabook?

The Dynabook after Xerox PARC

Alan Kay went on sabbatical from PARC in 1979 and never came back. The period from 1979 to 1984 would witness a mass exodus from Xerox PARC (Hiltzik 1999 describes at length the complex dynamics leading to this). While Kay's former colleagues, under Adele Goldberg's leadership, worked to prepare Smalltalk for an audience beyond PARC, Kay took the opportunity to become chief scientist at Atari, which in the early 1980s was the rising star in the nascent video game industry.

Kay spent four years at Atari, setting up research projects with long-range (7–10 year) mandates, but has described his role there as a "trojan horse." Inside each video game machine is a computer, and therein lies the potential to go beyond the video game. This period at Atari is represented in the literature more as a collection of war stories and anecdotes (see Rheingold 1985; Stone 1995) than of significant contributions from Kay himself.

A few important characters emerged out of Kay's team at Atari: notably Brenda Laurel, who went on to be a leading user-interface design theorist (Laurel & Mountford 1990; Laurel 1993) and later head of Purple Moon, a software company that targeted adolescent girls; and Ann Marion, whose "Aquarium" simulation project at Atari became the prototype for the research project which would define Kay's next phase. Ultimately, though, nothing of educational significance came out of Kay's term at Atari, and in 1984, corporate troubles ended his time there.

In late 1985 Kay took a research fellowship at Apple Computer that—along with the patronage of new CEO John Scully—seems to have given him a great deal of personal freedom to pursue his educational ideas. Kay stayed at Apple for a decade, firmly establishing a link between him and the popular company. Apple had become known for its emphasis on educational markets, which included large-scale donations of equipment to schools and research programs like "Apple Classrooms of Tomorrow" (Apple Computer 1995). Apple's presence in the education sector was key to their branding in the 1980s (as it remains today).

When Kay arrived, Apple had just released the first-generation Macintosh computer, the culmination of the work that had been inspired by the famous visit to Xerox PARC in 1979. The Mac was positioned as the alternative to the paradigm of personal computing defined by IBM's PC; the Mac was branded as "the computer for the rest of us." It was certainly the closest thing to the PARC genre of computing that the general public had seen. That the Mac was indeed different needs no re-telling here; the cultural and marketing battle between Macs and PCs (originally inscribed as Apple vs. IBM, later Apple vs. Microsoft) was the dominant metanarrative of 1980s computing. But despite the Mac's mouse-and-windows direct manipulation interface, it remained a long way from the kind of personal computing Kay's team had in mind (and had been literally working with) in the mid 1970s. The "look and feel" was similar, but there was no facility for the user shaping her own tools; nor was the Mac intended to be part of a network of users, as in the PARC vision. Nevertheless, Kay's oft-quoted pronouncement was that the Mac was the first personal computer "good enough to critique." And, as evidenced by the number of Kay's colleagues

who came to work at Apple in the 1980s, it must have seemed that the company was headed in the right direction.[16]

Kay's first year at Apple seems to have been spent writing, furthering his thinking about education and computing and projects. An article Kay had published in *Scientific American* (Kay 1984) gives a sense of where his thinking was. One of the key innovations of the microcomputer revolution—and the first really important answer to the "what are they good for" question—was a software application called *Visicalc*, the first dynamic spreadsheet program, introduced in the late 1970s by Dan Bricklin and Bob Franston. The spreadsheet is an interesting example of a computing application that was born on microcomputers; it is significantly absent from the lengthy collection of innovations from Xerox PARC, and PARC designers were very impressed when they saw it (Hiltzik 1999, p. 357). The spreadsheet concept clearly impressed Kay, too, and he framed it as a key piece of end-user empowerment:

> The dynamic spreadsheet is a good example of such a tissuelike superobject. It is a simulation kit, and it provides a remarkable degree of direct leverage. Spreadsheets at their best combine the genres established in the 1970s (objects, windows, what-you-see-is-what-you-get editing and goal-seeking retrieval) into a "better old thing" that is likely to be one of the "almost new things" for the mainstream designs of the next few years. (Kay 1984, p. 6)

On Spreadsheets

The spreadsheet program remains one of the most powerful—and flexible—tools on a personal computer. The original *Visicalc* was eclipsed by *Lotus 1-2-3* in the early 1980s, a program which made its developers into some of the first software multi-millionaires. In 1985, Microsoft released *Excel* as a spreadsheet for the Macintosh, blending the spreadsheet concept with a GUI. Excel today is ubiquitous and has no serious competition. However, the range of tasks to which Excel is put—far beyond general ledger accounting—is vast, including lots of small-scale end-user programming and personal data management.

A spreadsheet program like Excel is far more flexible and adaptable to a user's needs than a word processor. Interestingly, though, the basic way a spreadsheet program works hasn't really changed much in 25 years; the new versions have a lot of features, but the basic concept of a scriptable, two-dimensional grid of dynamically updated values remains.

16. Core members of Alan Kay's team—Larry Tesler, Ted Kaehler, and Dan Ingalls—went to Apple Computer in the early 1980s. Kay himself moved to Atari in 1980 and then Apple in 1984. Other ex-Xerox personalities spread to other key IT companies: word-processing pioneers Charles Simonyi and Gary Starkweather went to Microsoft, as did Alto designers Chuck Thacker and Butler Lampson. John Warnock and Charles Geschke, who worked on laser printing and the foundations of desktop publishing founded Adobe Systems. Bob Metcalf, who invented ethernet, founded 3Com.

Kay and his colleagues had come to recognize that the microcomputer was a serious force in the development of personal computing and not just a hobbyist's niche. The extent of Kay's engagement with the spreadsheet idea shows, if nothing else, that a current of new ideas from outside sources were a welcome addition, after a decade of research within PARC.

The Vivarium Project

Alan Kay's work at Apple was characterized by a single-minded return to the problem of how to bring scientific literacy to children via computing. His work on Smalltalk was over, and the corporate politics of Xerox were long gone. Kay took up his research fellowship at Apple by dedicating himself to working with kids again, something which had practically eluded him since the mid 1970s at Xerox PARC.

The project which most defined Kay's tenure at Apple through the mid and late 1980s and into the early 1990s was the *Vivarium*: a holistic experiment in technology integration that is possibly unparalleled in its scope. Kay's team moved in to the Los Angeles Open School for Individualization—LA's first "magnet school"—and stayed there for seven years. The scale of Apple's investment of time and resources in the school, and the Open School's contributions to Kay's research, make the Vivarium a very special educational technology project. Though little has been written about the Vivarium—compared, say, with the Apple Classrooms of Tomorrow (ACOT) program, which had a much higher public profile—it has had a lasting impact on Kay's work, educational computing research, and, of course, the LA Open School.

Ann Marion was the project's manager, and the basic idea for the project had come from her Masters thesis, written while working with Kay's team at Atari: to build a game from semi-autonomous cartoon characters. She wrote extensively about the project in a summative report for Apple entitled *Playground Paper* (Marion 1993). Marion decided on an ecological setting, with fish interacting with (e.g., eating) one another, and this became the core storyline behind the ambitious educational program at the Los Angeles Open

School. Within an explicitly progressivist and flexible educational setting at the Open School, Vivarium put the design and development of a complex ecological simulation in primary-school children's hands. The challenge for the kids was to create more realistic interactions among the fish while learning about ecological models along the way; the challenge for Kay's team was to extend computing technology to the children so that they could effectively carry this out. Larry Yaeger, one of the team members, later wrote:

> The literal definition of a "Vivarium" is an enclosure or reserve for keeping plants and animals alive in their natural habitat in order to observe and study them. The Apple Vivarium program is a long-range research program with the goal of improving the use of computers. By researching and building the many tools necessary to implement a functioning computer vivarium, an ecology-in-the-computer, we hope to shed light on many aspects of both the computer's user interface and the underlying computational metaphor. We are exploring new possibilities in computer graphics, user interfaces, operating systems, programming languages, and artificial intelligence. By working closely with young children, and learning from their intuitive responses to our system's interface and behavior, we hope to evolve a system whose simplicity and ease of use will enable more people to tailor their computer's behavior to meet their own needs and desires. We would like untrained elementary school children and octogenarians to be able to make specific demands of their computer systems on a par with what today requires a well trained computer programmer to implement. (Yaeger 1989)

The project was as broad-based as it was ambitious; it boasted an advisory board composed of various luminaries from the world of cognitive science, *Hitchiker's Guide to the Galaxy* author Douglas Adams, and even Koko, the famous gorilla (at one point, the team was engaged in creating a computer interface for Koko). The computer-based work was enmeshed in a much larger, exploratory learning environment (the school featured extensive outdoor gardens that the children tended) at the Open School. Yaeger wrote:

> The main research test site of the Vivarium program is a Los Angeles "magnet" school known as the Open School. Alan chose this primary school, grades 1 through 6 (ages 6 through 12), because of their educational philosophy,

> founded on the basic premise that children are natural learners and that growth is developmental. Based on Piaget's stages of cognitive development and Bruner's educational tenets, the Open School was seen not as an institution in need of saving, but as an already strong educational resource whose fundamental philosophies aligned with our own. With the support of the Open School's staff, some 300 culturally and racially mixed children, and our principal liaison with the school, Dave Mintz, we have developed an evolving Vivarium program that is included in their Los Angeles Unified Public Schools curriculum. (Yaeger, 1989)

The LA Open School had been established in 1977, "by a group of parents and teachers who wanted an alternative to the 'back-to-basics' approach that dominated the district at that time. The group wanted to start a school based on the principles of Jerome Bruner and the practices of the British infant schools" (SRI International 1995). It was the LA Unified School District's (LAUSD) first magnet school, mandated to pursue a highly progressive agenda, with few of the restrictions that district schools worked within. The school held 384 students (K–5) and 12 teachers, arranged in 2-year multigraded "clusters," team-taught by two teachers with 62 kids in each (BJ Allen-Conn, personal communication, Nov 2004). Ann Marion characterized the school setting:

> The L.A. school we chose to work with we believed had no need of "saving." The Open School for Individualization, a public magnet school in Los Angeles, emphasizes theme-based projects around which children learn by bringing all the classroom subjects together in service of the theme. Construction projects evoke a whole person approach to learning which engages many different mentalitites. In this regard we share the common influence of Jerome Bruner which is evident throughout these different activities. We find models of group work. Variety and effectiveness of working groups are to be seen of different sizes, abilities and ages, in which children collaborate and confront each other. (Marion 1993, ch 2, p. 1)

By 1985—before Apple's involvement—the school had apparently already begun to integrate microcomputers, and the teachers had used Logo. At that time there was already a strong notion of how computers should be used: in the classroom, not in labs, and for crea-

tive work, as opposed to drill-and-practice work or games (Allen-Conn, personal communication). In that year, Alan Kay had contacted the Board of Education looking for a school to serve as a research bed; he had also inquired at the Museum of Science and Industry, and made a list of possible schools. After visiting several, Kay felt that the Open School was philosophically closest to what he had in mind, that the school was indeed the right kind of environment for his research. Originally, Kay and Marion planned to run the Vivarium in one classroom, but principal Bobby (Roberta) Blatt insisted that any resources be used for the entire school; that it would disrupt the democratic and consensual nature of the school to have one class with an inordinate amount of technology. Kay's research could focus on one classroom (it largely did), but the resources had to be managed across the whole school (Bobby Blatt, personal communication, Nov 2004).

Blatt, the teachers, and the parents at the Open School negotiated extensively with Kay to establish the terms of the relationship; while they were open and excited by the possibilities of using technology intensively, they were concerned that Apple's involvement would change the "tenor" of the school (Blatt, personal communication); instead, they wanted the technology to be "invisible" and fully integrated into the constructivist curriculum they were creating. The focus had to be on the children and their creations. For instance, the school had a policy against video games on the school's computers—*unless the children themselves created the games*. Blatt reports that this spawned a culture of creating and sharing games, mostly developed in Apple's *HyperCard* authoring software. In January 1986, when the Vivarium project was launched, Apple Computer installed one computer per two children,[17] began training the teachers and staff and provided a technical support staffperson at the school.

The Open School also negotiated an investment by Apple into maintaining the school's arts, music, and physical education curriculum—since arts curriculum funding was under the axe in California. Kay was more than happy to comply with this, and so an investment

17. The computers at the LA Open School were installed inside the desks, and the desktops replaced with a piece of plexiglass. This allowed the computers to be installed in regular classrooms without making the classroom desks useless for any other use. (see Kay 1991)

on the order of $100,000 was made annually to the Open School for curriculum and activities that had nothing directly to do with computing, but served to keep academics, arts, and technology in balance—and which also provided time for the core teaching staff at the Open School to do group planning. Both Blatt and long-term teacher BJ Allen-Conn (personal communication, Nov 2004) reported that once everyone got to know Kay, their fears of being overwhelmed by the technological agenda quickly abated, owing to Kay's "respectful attitude." Ongoing collaboration between Kay's group and the Open School teachers took place at monthly "brown bag" lunches and teas.

By the late 1980s, the twelve teachers at the Open School were working part-time with Kay's team—10 hours per week plus 6–10 weeks per summer, as well as conferences and other professional-development events through the year (Marion 1993, ch. 2, p. 10)—and being paid consulting fees to help develop curriculum for themes at the Open School. Kim Rose claims that some of the teachers bowed out after a few years of this, simply because they wanted to have regular summer vacation for a change (Rose, personal communication, Oct 2004), but this general pattern continued right into the early 1990s.

What is obvious here, but which bears dwelling upon for a moment, is that the relationship between Kay's team at Apple and the LA Open School was an unprecedented and unparalleled situation of support, funding, and devotion to pursuing the ends of technology integration. It is hard to imagine any other school situation even comparable to this. But rather than thinking of the Vivarium project at the Open School as any sort of model for technology integration, we should consider the Open School part of Kay's research lab. At Xerox PARC, it had been a challenge to sustain access to children and school settings, or conversely to sustain children's access to Xerox labs. At Apple, Kay's agenda seems to have been to set up a rich and ongoing relationship with a school as a foundational element of the project, and then to move forward with the technological research—somewhat the reverse of the arrangement at Xerox PARC.

The Vivarium project itself had a broad and holistic vision; it modelled both an exploratory educational vision and a way of integrating technology with education. Apple

management seems to have given Kay the room to pursue a pure research agenda, but this statement needs qualification: the Open School was in every way a 'real' and applied setting. Rather, Kay colleagues' research there seems to have had little impact on Apple's products or its ostensible presence in the marketplace and the 'purity' of the research should be seen on this corporate level rather than the decidely messy pedagogical level. Ann Marion's summative report on the project goes into some detail about the practical difficulty of attempting to keep the technology subservient to curriculum ends. But Vivarium was a low-profile 'skunkworks' research project, interested in furthering blue-sky research into a wide variety of computing themes—simulation environments, animation systems, user-interface techniques (both hardware and software)—in comparison with the much higher-profile "Apple Classrooms of Tomorrow" program, which sought to deploy existing Apple technologies to schools and to be focused on "technology transfer" (Ann Marion, personal communication, Nov 2004).

The Vivarium project had run out of steam by 1993, when Apple fell on hard times economically and organizationally (Kay's patron, John Sculley, was ousted as CEO in 1993). Bobby Blatt reports that Apple's pullout from the LA Open School was conducted with lots of advance warning, and that the parents were motivated to keep the same level of computer integration at the school, having "tasted the wine" (Blatt, personal communication), a challenge which they have appearently succeeded at. In 1993, principal Blatt, looking at retirement and anticipating Apple's withdrawal from the school, pursued Charter School status for the LA Open School, which would ensure its continued autonomy; she succeeded, and it became the Open Charter School in 1994, continuing along the same lines today. Apple Computer's investment of people and research has not been duplicated. However, Kay's own team has maintained some level of involvement with the Open School (and in particular, with teacher BJ Allen-Conn) ever since. In fact, the foundation of a research group that would provide Kay's working context for the next decade was by this point established. Kim Rose, for instance, was hired on to the Vivarium project in 1986, and remains Kay's closest working partner today.

Vivarium research

The research conducted through the Vivarium years seems to have two facets: the first, with the Open School in mind, was the creation of a simulation environment (of an underwater ecology) in which primary-school kids could act as designers and systems modellers as they developed their understanding of ecosystem dynamics. The second, which potentially had more application to Apple's own agenda, was an investigation of end-user programming, with a definition of "end-user" beyond the children at the LA Open School.

The simulation research drew largely on the work Kay's team had done with Smalltalk while at Xerox; the essential challenge is to figure out what sort of basic scaffolding will allow children to work at the level of *design* and problem-solving rather than wrestling with the syntax and mechanics of the environment (Kay & Goldberg 1976; Goldberg 1979; Goldberg & Ross 1981; Marion 1993). The Vivarium project engaged several teams of developers (often drawn from the MIT Media Lab) to try out various approaches to this challenge. Mike Travers' MS Thesis from MIT, entitled "Agar: An Animal Construction Kit" (1988) was the result of one early project. Agar provided a customizable, agent/rules-based environment for setting up autonomous virtual actors and scripting their prototypical reactions to one another. Another example is Jamie Fenton and Kent Beck's first-generation "Playground: An Object Oriented Simulation System with Agent Rules for Children of All Ages" (1989); *Playground*—in the Fenton and Beck version and in subsequent versions developed by Scott Wallace—was a more abitious agent/rules system which prototyped a scripting language and environment designed for children to describe interrelationships between actors. There were numerous other prototypes, including icon-based graphical programming environments and a variety of other ideas. The Playground system eventually emerged as the dominant platform for the simulation projects at the Open School. Ann Marion characterized it thus:

> The playground was chosen as our metaphor for a computer programming environment ... The playground is a place where rule-governed activities have a natural place, involving play, invention, and simulation. On the playground,

> children assume roles which limit their behavior to that of defined and shared characters. Rules and relationships are endlessly debated and changed. The nature and structure of playground play resembles some of the strategy children might exercise on the computer, to set up computer instructions in construction and play with simulations of multiple players. (Marion 1993, preface, p. 3.)
>
> One way to think about Playground is as having a spreadsheet view, a HyperCard view, and a textual programming view, all simultaneously available, where the user can make changes in whatever view seems easiest to work with, and have all views updated appropriately. (ch 1. p. 1)

A point which is easy to overlook from the vantage point of the 21st century is the sheer challenge of making systems of this sophistication workable on the computers of the mid 1980s. The later versions of the Playground software were developed using a version of Smalltalk for the Macintosh. This might seem like a straightforward thing to do, given the historical sequence, but performance limitations of 1980s-era Macintosh computers meant that this must have been a constant headache for the team; what had been possible on expensive, custom-designed hardware at Xerox was not nearly as practical on relatively inexpensive Macs, even a decade later. At one point, a special Smalltalk accelerator circuitboard had to be installed in the Macs at the Open School to get an acceptable level of performance in Playground. In a sense, the Vivarium project can be seen as a massive logistical challenge for Kay: how to move from a context in which all the technical facets are (reasonably speaking) within his team's control to one where one's ideas are constantly running up against basic implementation obstacles. At the same time, of course, the engagement with the Open School and the children there was far deeper and longer-term than anything Kay had experienced at Xerox.

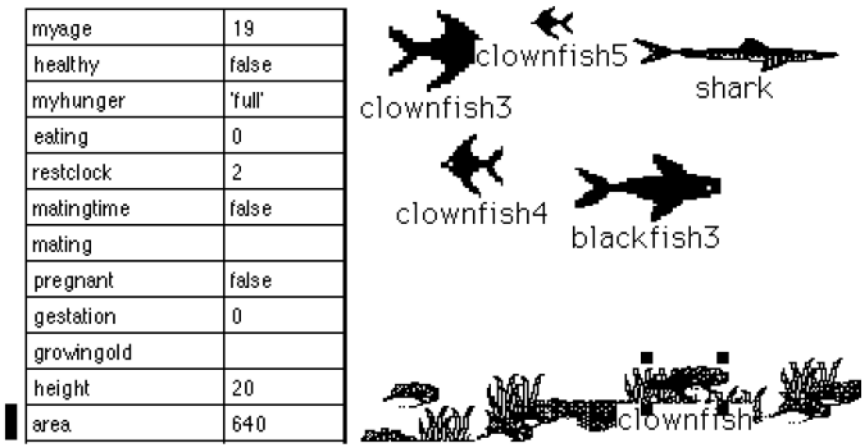


Figure 5.6: Playground environment, circa 1990 (from Marion 1993)

The other research avenue, into end-user programming as a general topic, is an example of Kay's body of well thought-out ideas coming in contact with a wealth of related ideas from others and other contexts. Clearly, after a decade of work with Smalltalk, and having had the opportunity to define much of the problem space (of how personal computing would be done) from scratch, Kay and his colleagues from PARC had done a huge amount of thinking already. Within Apple Computer, though, were a number of people who had come at the topic of end-user programming from different perspectives. An Apple "Advanced Technology Research Note" from the early 1990s (Chesley et al. 1994) reveals a rich and fecund discourse going on within Apple, despite a relative dearth of results being released to the PC marketplace and computer-buying public. Apart from the already-mentioned spreadsheet model, the standout example was HyperCard. Kay and his team had the opportunity to engage with and learn from some substantial development efforts, and to watch how real end-users—both children and adults (teachers among them)—reacted to various systems.

What must be underscored in this discussion about Alan Kay's tenure at Apple Computer is that despite the low profile of the Vivarium project[18] and the dearth of overt outcomes (software, publications, further research programs), Kay must be credited with sticking tightly to his agenda, resisting the translation of the project into other trajectories, be they corporate or technical, as had happened at Xerox. In short, Kay appears to have fought to keep the black boxes open at all costs (Ann Marion's report supports this). Ultimately, though, by the mid 1990s, with the Vivarium project wound up and Apple in corporate trouble, Kay found himself with rather less social capital than he would have liked; without over-dramatizing too much, we might conclude that Kay's efforts to keep his research project "pure" risked its ongoing support at Apple—and perhaps beyond. The ultimate outcomes of the Vivarium project had impacts on the persons involved, and we shall see how this affects Kay's subsequent work, but it is hard to see the broader educational (or even technical) results—to say nothing of the influence—of Vivarium. This observation is, I believe, in line with the general sense of Latour and Callon's network theory: it is in being translated that a project or system gains greater currency and interconnectedness. A project which remains tightly defined perhaps does so at the expense of its larger reach.

Message-passing vs. Value-pulling

The *Playground* environment (developed twice: in the late 1980s by Jay Fenton and Kent Beck, and in the early 1990s by Scott Wallace) ironically represents a conceptual reversal of the computing model Kay had pioneered in the 1970s.

Kay's fundamental contribution to computer science (via Smalltalk) is the centrality of message-passing objects. The term is "object orientation," but Kay has noted that the more important concept is that of *message passing*. In a 1998 mailing-list posting, Kay clarified:

> *The Japanese have a small word—ma—for "that which is in between"—perhaps the nearest English equivalent is "interstitial." The key in making great and growable systems is much more to design how its modules communicate rather than what their internal properties and behaviors should be. (Kay 1998b)*

Playground, however, completely eschewed message-passing. Inspired by the spreadsheet model, Playground was made up of objects which had various parameters. The values of these parameters could then be *looked up*, and chains of effect could be built out of such retrievals (just like in a complex spreadsheet). Instead of an object actively sending a mesage to another object, objects continually *watched* for changes in other objects, just as a dynamic formula cell in a spreadsheet watches for changes in the data cells it takes as input. When the data cells change, the dynamic formula updates its own value.

The architecture was perhaps apt—Playground was designed for a marine ecosystem simulation, and, arguably, plants and animals do more observation—"noticing"—of each other's states than sending messages (Ted Kaehler, personal communication, Oct 2004). Playground provided a wealth of insights, but proved difficult to program complex behaviour. Later software projects returned to the message-passing model, combined with pieces of the dynamic-retrieval model.

HyperCard and the Fate of End-User Programming

In assessing the fate of Kay's projects and ideas while at Apple, it is instructive to consider the contemporaneous example of HyperCard, a piece of personal media software similar in spirit to parts of the Dynabook vision. HyperCard was relatively successful in comparison, but had a decidedly different history and aesthetic. Like the dynamic spreadsheet, HyperCard forced Kay and his team to take notice, and to look very closely at its success. HyperCard's ultimate fate, however, points to larger cultural-historical trends which significantly affected Kay's projects.

Translation #6: From media environment to "Multimedia Applications"

HyperCard was a pet project of Bill Atkinson, who was perhaps the key software architect of the original Macintosh and its graphic interface (he had been part of the team from Apple that went to Xerox PARC to see Smalltalk in 1979). Atkinson's place in Macintosh history was further cemented with his release of *MacPaint*, the original Macintosh graphics application and the ancestor of software like Adobe Photoshop. After MacPaint, Atkinson began playing with the notion of an interactive presentation tool called *WildCard*. WildCard—rebranded *HyperCard* in 1985—was based on the metaphor of a stack of index cards which could contain any combination of graphic elements, text, and interactive buttons. The main function of such interactive buttons was to flip from one 'card' view to another, thereby making HyperCard a simple hypermedia authoring tool. HyperCard put together an unprecedented set of features—graphics tools like MacPaint, a simple text editor, support for audio, and, with the addition of a scripting language called *HyperTalk* in 1987, a simple and elegant end-user programming environment.

Despite what the historical sequence might suggest, and despite some similarities which appear in HyperCard (a rough object model, message passing, and the "HyperTalk"

18. The published literature on the Vivarium is very thin, given that the project ran for almost a decade. Notable is an article Kay wrote for *Scientific American* in 1991, "Computers, Networks and Education," which is heavy on Kay's educational philosophy and light on project details. The project gets a brief mention in Stewart Brand's popular book, *The Media Lab: Inventing the Future at MIT* (1987), due to the involvement of several Media Lab researchers.

scripting language), Smalltalk was *not* a direct influence on HyperCard; rather, it apparently came more-or-less fully formed from Bill Atkinson's imagination. Atkinson had of course seen Smalltalk, and there were notable ex-PARC people including Alan Kay and Ted Kaehler at Apple (and even involved in HyperCard's development), but HyperCard and its workings were Atkinson's own (Ted Kaehler, personal communication, July 2004).

HyperCard's great innovation was that it brought the concept of hypermedia authoring down to earth; it was the first system for designing and creating non-linear presentations that was within the reach of the average PC user. A designer could put any combination of media elements on a given card, and then create behaviours which would allow a user to move between cards. The system was simple to grasp, and in practice, proved easy for users of all ages to create "stacks," as HyperCard documents were called.

Key to HyperCard's success was Apple's decision to pre-install HyperCard on all new Macintosh computers after 1987. The result was a large HyperCard community that distributed and exchanged thousands of user-created Hypercard stacks, many of which took the form of curriculum resources for classrooms.[19] Alternatively, HyperCard was seen as a multimedia authoring toolkit and was put to use as a writing and design medium (or multimedium, as it were), again, often in classrooms; the genre of multimedia "authoring" was first established in this period, indicating the design and construction of hypermedia documents in a tool such as HyperCard. Ambron & Hooper's 1990 book, *Learning with Interactive Multimedia*, is a snapshot of the kinds of uses to which HyperCard was being put in the late 1980s.

Not surprisingly, HyperCard was introduced very early on to the teachers and staff at the Open School, and met with considerable zeal; the teachers there could quickly see applications for it, and could quickly figure out how to realize these. The children were able to work with it easliy, too. This experience was in some contrast with Playground and the simulation environments, which, although being much more sophisticated, were barely

19. HyperCard was used early on to control a videodisc player attached to one's Macintosh; this gave HyperCard the ability to integrate large amounts of high-quality multimedia content: colour images and video, for instance.

usable on the limited hardware of the day. This, in combination with HyperCard's elegant balance of simplicity and flexibility, proved to be a lesson Kay took to heart; here was a system that managed to acheive the low threshold of initial complexity that Kay had been shooting for over a decade or more.

Still, HyperCard's limitations frustrated Kay. As elegant in conception and usability as it was, HyperCard was nowhere near the holistic media environment that Smalltalk had been. And while Kay praised HyperCard for its style and its obvious appeal to users, he railed against its limited conceptual structures: "That wonderful system, HyperCard, in spite of its great ideas, has some 'metaphors' that set my teeth on edge. Four of them are 'stack,' 'card,' 'field,' and 'button'" (Kay 1990, p. 200)—the entirety of HyperCard's object model! This is not mere griping or sour grapes on Kay's part; the object paradigm that Smalltalk pioneered meant that objects were fundamental building blocks for an unlimited range of conceptual structures; to restrict a system to four pre-defined objects misses the entire point. That said, that HyperCard in its conceptual simplicity was an immediate success—not just at the Open School, but with users around the world—was not lost on anyone, least of all Kay, and its influence would be felt in his later work.

HyperCard's limitations were felt by others, too. It became clear that, though HyperCard could do animation, a dedicated tool like VideoWorks (the prototype for MacroMedia's "Director" software) was a better animation tool. Similarly, MacPaint and like graphics programs were more flexible than HyperCard (which, despite its decade-long life, never went beyond black-and-white graphics). The very idea of an all-encompassing media environment like Smalltalk, and, to a lesser extent, HyperCard, was doomed to buck the trend toward a genre of discrete application programs: individual word processors, spreadsheets, paint programs, and so on. If nothing else, a dedicated tool like a paint program was easier to market than an open-ended authoring environment. The "what are they good for" question is directly the point here. This drawing program is excellent for making diagrams; this word processor is excellent for writing letters; but what was HyperCard good for, exactly? Its hundreds of thousands of users all had an idea, but it is important

to remember that none of these early users had to make a purchasing decision for HyperCard, since it had been given away with new Macs. Even HyperCard ultimately had to justify its existence at *Claris*, the software company spun off from Apple in the early 1990s, by claiming to be a "multimedia" toolkit, and was branded by Apple as a "viewer" application for HyperCard stacks; the authoring functionality was sold separately. HyperCard was ultimately abandoned in the 1990s.

HyperCard's relative success in the personal computing world underscores two points, which, seen through the lenses of Latour's translations and subsequent black boxes, appear thus: first is the shift from media environment to "application toolkit," with the attendant metaphors: standardized palettes, toolbars, and the establishment of the commodity "application" as the fundamental unit of personal computing. The second shift is from a generalized media environment to that of "Multimedia" applications, reifying "Multimedia" as an industry buzzword. As these concepts become commoditized and reified in the marketplace, the interesting work of defining them shifts elsewhere.

Translation #7: From epistemological tools to "Logo-as-Latin"

Seymour Papert's work with the Logo programming language had begun as early as 1968, but despite its significant impact on Alan Kay's personal mission and despite a good number of published articles from the early 1970s, Logo made very little impact on the public imagination until 1980, with the publication of Papert's signature work, *Mindstorms: Children, Computers, and Powerful Ideas*. The appearance of this book set the stage for a significant commercialization and marketing effort aimed at getting Logo onto the new personal microcomputers. Programming in Logo grew into a popular computing genre through the early 1980s. A look at library holdings in the LB1028.5[20] range reveals a huge surge of output surrounding Logo in the classroom in the mid 1980s. Papert and Logo had become practically synonymous with educational technology in these years. But of course any substantial movement of an idea—let alone a technological system—into very different and (and vastly

20. LB1028.5 is listed in the Library of Congress Classification as "Computer assisted instruction. Programmed instruction"—ironic, given Papert's comments on children programmming computers and vice-versa.

larger) contexts brings with it necessary translations. In the case of Logo, this shift was in the form of *branding*. *What was Logo*, that it could be rapidly picked up and spread across school systems in North America and Europe in just a few short years (Aglianos, Noss, & Whitty 2001)? Chakraborty et al. (1999) suggest that the effort to make Logo into a marketable commodity effectively split the Logo community into "revolutionists" like Papert, interested in a radical redefinition of mathematics pedagogy, and more moderate "reformers," who were more interested in spreading Logo as widely as possible.

This means that *what Logo became* in the marketplace (in the broad sense of the word) was a particular black box: turtle geometry; the notion that computer programming encourages a particular kind of thinking; that programming in Logo somehow symbolizes "computer literacy." These notions are all very dubious—Logo is capable of vastly more than turtle graphics; the 'thinking skills' strategy was never part of Papert's vocabulary; and to equate a particular activity like Logo programming with computer literacy is the equivalent of saying that (English) literacy can be reduced to reading newspaper articles—but these are the terms by which Logo became a mass phenomenon. Papert, for better or worse, stuck by Logo all the while, fighting something of a rear-guard action to maintain the complex and challenging intellectual foundation which he had attempted to lay. It was perhaps inevitable, as Papert himself notes (1987), that after such unrestrained enthusiasm, there would come a backlash. It was also perhaps inevitable given the weight that was put on it: Logo had come, within educational circles, to *represent* computer programming in the large, despite Papert's frequent and eloquent statements about Logo's role as an epistemological resource for thinking about mathematics. In the spirit of the larger project of cultural history that I am attempting here, I want to keep the emphasis on what Logo *represented* to various constituencies, rather than appealing to a body of literature that reported how Logo 'didn't work as promised,' as many have done (e.g., Sloan 1985; Pea & Sheingold 1987). The latter, I believe, can only be evaluated in terms of this cultural history.

Papert indeed found himself searching for higher ground, as he accused Logo's growing numbers of critics of *technocentrism*:

> Egocentrism for Piaget does not mean "selfishness"—it means that the child has difficulty understanding anything independently of the self. Technocentrism refers to the tendency to give a similar centrality to a technical object—for example computers or Logo. This tendency shows up in questions like "What is THE effect of THE computer on cognitive development?" or "Does Logo work?" ... such turns of phrase often betray a tendency to think of "computers" and "Logo" as agents that act directly on thinking and learning; they betray a tendency to reduce what are really the most important components of educational situtations—people and cultures—to a secondary, faciltiating role. The context for human development is always a culture, never an isolated technology. (Papert 1987, p. 23)

But by 1990, the damage was done: Logo's image became that of a has-been technology, and its black boxes closed: in a 1996 framing of the field of educational technology, Timothy Koschmann named "Logo-as-Latin" a *past paradigm* of educational computing. The blunt idea that "programming" was an activity which could lead to "higher order thinking skills" (or not, as it were) had obviated Papert's rich and subtle vision of an ego-syntonic mathematics.

By the early 1990s, the literature on educational technology had shifted; new titles in the LB1028.5 section were scarce, as new call numbers (and thus new genres) were in vogue: instructional design (LB1028.38); topics in the use of office productivity software (LB1028.46) and multimedia in the classroom (LB1028.55). Logo—and with it, programming—had faded. This had obvious effects for other systems—like HyperCard (Ted Kaehler, personal communciation). In fact, HyperCard's rise to relative popularity in this same period (and in similar call numbers) is probably *despite* its having a "programming" component; its multimedia strengths carried it through the contemporary trend. To my knowledge, there is no scholarship tracing the details of HyperCard's educational use historically, but one piece of evidence is a popular competitor (perhaps it would be better to say "successor") to HyperCard called *HyperStudio*. HyperStudio featured roughly the same stack-and-cards metaphor, and added colour graphics, but dropped HyperCard's elegant scripting language. In fact, and somewhat ironically, later releases of HyperStudio incorpo-

rated a language called "HyperLogo" (perhaps to flesh out the program's feature list), though it was not particularly well integrated,[21] and there is little evidence that it made much of an impact on HyperStudio's use.

Similarly, a new genre of simulation environments for teaching systems concepts (e.g., *SimCalc*) eschewed the notion of 'whole' environments, prefering instead to provide neatly contained microworlds with a minimum of dynamic scope; these are obviously quicker to pick up and easier to integrate into existing curriculum and existing systems.[22]

The message—or black box—resulting from the rise and fall of Logo seems to have been the notion that "programming" is over-rated and esoteric, more properly relegated to the ash-heap of ed-tech history, just as in the analogy with Latin. Moreover, with the coming of "multimedia" as the big news in early-1990s educational computing, the conclusion had seemingly been drawn that programming is antithetical to 'user-friendliness' or transparency. How far we had come from the Dynabook vision, or any kind of rich notion of computational literacy, as diSessa called it:

> The hidden metaphor behind transparency—that seeing is understanding—is at loggerheads with literacy. It is the opposite of how media make us smarter. Media don't present an unadulterated "picture" of the problem we want to solve, but have their fundamental advantage in providing a *different* representation, with different emphases and different operational possibilities than "seeing and directly manipulating." (diSessa 2000, p. 225)

The Dynabook vision seemed further away than ever! Smalltalk, no matter what you may call it, is a programming language. *Or is it?* To answer that question, we first need a more comprehensive assessment of what personal computing means to us today.

---

21. I had the opportunity to write high-school Information Technology curriculum materials for distance education in the late 1990s. HyperStudio was a popular resource and I was encouraged to write it into my materials. However, the HyperLogo implementation so underwhelmed me that I rejected it in favour of a plain and simple Logo implementation (UCBLogo) for a module on introductory programming.

22. Interestingly, Jeremy Roschelle and colleagues on the *SimCalc* project argue persuasively for a "component architecture" approach as an alternative to all-encompassing systems (Roschelle et al. 1998), but this must be read in historical context, appearing in a time when networking technology was re-appearing as a fundamental component of personal computing and monolithic application software was being challenged.

Chapter 6:
Personal Computing in the Age of the Web

What is a "Powerful Idea," Anyway?

There was a time in educational computing—the time we have been speaking of so far—when talk of "powerful ideas" was the order of the day. But today, there is no mention of powerful ideas—or ideas at all, for that matter—in discussions of learning management systems, interoperability standards, or test banks. To what can we attribute this shift? Is it merely that the early exuberance of pioneers like Kay and Papert has given way to a more sober, mature perspective on the daily business of education? Is it that we have now seen through the "technohype," realizing, after all, that we were the dupes of salespeople and other evangelists? Was all that talk of powerful ideas just marketing? Just what is a powerful idea, anyway?

Powerful ideas don't come out of thin air; they have histories, and politics. Papert pointed out—in what is no doubt something of a reponse to the *Logo-as-Latin* eulogy—that Latin was not so long ago equated with "powerful ideas;" that it was commonly felt that learning Latin made students think more logically, or, in today's parlance, that it "fosters higher-order thinking skills."[1] But we think such ideas quaint today. The alternative to "Latinesque" curriculum, Papert says, is "Driveresque"— that is, oriented to clear practicalities. Not quaint, to be sure.

But where do we place math curriculum along this continuum? Solving quadratic equations hardly counts as "Driveresque" for the vast majority of us. Does this mean that the teaching of such concepts as algebra is quaint? Or do we still believe there is some more general value in mathematics? Algebra is—in our time—taken as a *powerful idea*, and this justifies its general inclusion beyond what is merely practical in the context of graduates' day-to-day life skills. It is not so hard to point to other such powerful ideas lurking in school

1. Papert made these comments is a guest lecture in an undergraduate computing course taught by Alan Kay at UCLA, April 15, 2004. Papert's presentation that day was partially a reprise of his talk at the 2004 AERA conference a few days earlier.

curriculum: liberal democracy, the atomic theory of matter, supply-and-demand economics, cell biology, reading and writing, or Maxwell's equations regarding electromagnetic fields. In the United States currently, there is substantial debate over just how powerful an idea *evolution* is.

The latter example points to an important consideration: what *counts* as a powerful idea is something that is constructed, not given. What makes an idea "powerful" is what it allows you to do; in the vocabulary of the sociology of translation, powerful ideas are related to the lengthening of sociotechnical *networks*. Once you have a powerful idea established within the discourse, you now have access to the range of ideas—of articulations—connected to it. Powerful ideas are those that are richly or deeply connected to other ideas; these connections make it possible to make further connections and tell stories of greater richness and extent. Algebra is a powerful idea because of what it leads to: science, engineering, analysis, generalization of quantifyable patterns; without it, one is profoundly limited in access to these realms of understanding and agency. But algebra is not a powerful idea on its own. Nor was Latin during the Renaissance; rather, the teaching of Latin represented a powerful idea because of what classical literacy led to: a whole universe of discourse.[2]

Following on this, the relevance of these ideas to education is a matter of considerable importance, historically contingent and entirely political. An educational system that places central value on algebra or evolution or Latin is taking a particular political stance with respect to the accessibility of those networks of human activity. Such curriculum decisions are not made by appeal to their direct relevance to students' lives; there is very little immediate practical applicability for high-school students of, say, Maxwell's equations, but the connections this particular powerful idea leads to are of enormous scope. The extent to which educators present such ideas has significant political importance: who should be

2. The teaching of Latin is easy to dismiss as an idea whose relevance has simply faded; other 'powerful' ideas with great currency in the past have been criticized for more acute reasons: Intelligence Quotients and racial hierarchy are two prominent 20th-century examples. We will no doubt in the future reject a number of 'powerful' ideas on grounds of either their quaintness or our moral indignation.

taught something like Maxwell's equations? Only students planning to study science or engineering in University? Everyone? Just the boys? Just the girls?

Clearly, such a decision requires the weighing of many competing priorities: a model of electromagnetic waves is only one of many important concepts vying for limited curriculum time; curriculum must also address issues of available resources, teaching time, learner context, and a host of other practicalities. But my point here is not to argue for the importance of one or other concept, but to demonstrate that "powerful ideas"—though we might disagree about which ideas qualify—are of core importance in *everyone's* understanding of curriculum: officials, administrators, educators, parents, and students themselves. "Powerful ideas" do not operate in any mysterious way, for they are (merely) key pieces of contemporary worldviews. The issue of which particular ideas count as powerful in any particular context is enormously complex; but save the chronically cynical among us, I do not think anyone—however materialist, pragmatist, or, for that matter, postmodern they might be—would seriously claim to have outgrown or rejected powerful ideas in general and their role in shaping our worldviews.

So why have powerful ideas been evacuated from educational computing in the past decade or so? I do not mean just the particular "powerful ideas" espoused by Papert or Kay; there has in recent years been no discussion or, apparently, concern, with what the powerful ideas in computing and information technology might really be—save perhaps a vague and overgeneral regard of the Internet's enormous scale. We have instead apparently abdicated the task of thinking deeply about the importance—technically, politically, and ultimately morally (any truly powerful idea has a moral dimension)—of computing. Is this not the responsibility of education, of schooling?

To investigate this issue, and to perhaps come to a cultural/historical appreciation of the current malaise, we will need to take stock of the world of computing today, in the age of the Web.

The 1990s: The Arrival of the Web

I lead my analysis of the 'current' state of personal computing with a discussion of the World-Wide Web, for it is surely the defining point of (personal) computing today. The Web is, interestingly, based on technologies and ideas which pre-date personal computing. It is, in many ways, the realization of ARPA director JCR Licklider's vision (1960) of a global public information utility. It is the ARPA dream, finally writ large, with public access to time-shared[3] computing resources all over the world; the computing paradigm of the mid and late 1960s finally come to fruition twenty-five years after the fact. This little anachronism was perhaps the first and original compression of time and space that the "cyberculture" era would experience.

Nonetheless, the explosion of Internet access to a wider public in the 1990s was a very exciting thing, and there came with it a great wave of creative energy. With a critical mass of online users, the dynamics of popular IT began to shift. Software, for the first time for PC users, began to flow like a fluid, rather than being distributed (and purchased) in boxes. I recall new versions of Internet programs appearing every couple of weeks, with each one doing so much more than its predecessors that I had to re-adjust my conceptual goggles repeatedly.

In 1993, it was not at all obvious that the Web would be the next big thing. It was certainly not the only contender. At the time, the fledgling Web competed for attention with a variety of other comers: there emerged at roughly the same time a similar document retrieval application called Gopher, which allowed a browser to traverse hierarchical menus leading to information resources around the world. There was a growing collection of search and indexing tools for organizing and traversing a world of FTP-based resources. At the same time, there was a flurry of activity around online games and chat—the MUDs and MOO spaces I referred to earlier—which I believed to be more interesting than web pages.

3. The asymmetrical, client-server architecture of the Web is essentially that of a time-sharing application, despite the existence of sophisticated computers at the client end. The server side of a Web application slices its computing resources in time for a variety of roughly concurrent distributed users.

There were also the already venerable electronic mail and USENET discussion forums, which consumed the majority of Internet traffic (and probably user attention). But it would be the Web—invented in Switzerland in 1990 and propelled to popular appeal by the release of the University of Illinois' *Mosaic* web browser in 1993—that would see exponential growth through the 1990s. The Web quickly obliterated Gopher and the FTP-based way of distributing information and became the key Internet application: the "killer app" as Tim O'Reilly pointed out (O'Reilly 1999)—that is, the reason people wanted to get on the Internet, more important than e-mail, or discussion groups, or any of the novel ideas which were popping up. The Web gave the Internet a familar face—the 'page'—which new users could easily see and recognize. I recall talking with a friend—a fellow enthusiast—in early 1994, remarking that websites were really remarkably like *magazines*. I was not the only one to see that analogy.

Translation #7: From stand-alone PCs to information applicances

The resulting dynamic is pretty clear to anyone who has not been asleep for the past decade; the Web expanded exponentially, spawned an industry, and became such a dominant paradigm that it ended up stifling a good deal of the creative outflowing of the early 1990s. The industrialization of the Web has been nearly all-encompassing. The Web has been such a "killer app" that it is undoubtedly one of the key factors in the eventual realization of one of the missing pieces of the "PARC genre:" the networked PC. Before the advent of the Web, most PCs weren't networked at all; those that were interconnected lived in offices where the network allowed access to a particular resource: a shared printer, a file server, a database. After the Web, a network interface became an integral part of every PC manufactured. *What good is a PC without a network connection*, we can now legitimately ask? Hence, the personal computer as communications medium has finally been realized. The black boxes resulting from this shift are those of the networked PC as commodity and the magazine/brochure metaphor for the Web and (by extension) the Internet.

Despite the Web's obvious virtues as a mass medium, two large-scale trends have shadowed its growth as an actor in our daily information lives. The first is that user interface has been largely reduced to the lowest common denominator provided by Web browsers. Instead of a thriving, pluralistic ecology of interface designs (as was apparent in the early 1990s, especially as multimedia became a hot topic), there is today just one interface, that provided by the Web (even further: that provided by a single browser: Microsoft's Internet Explorer). That the user interface is now standardized, allowing people to negotiate new information spaces without having to concern themselves with the mechanics of the software, is of course not a bad thing, not in itself. The trouble is that we have settled for a Web interface that is, frankly, quite crude compared with the visions of personal computing, information access, and networked computing that preceded it. The Web offers nothing but a simple, stateless query-and-response file delivery service, and the page-description language HTML is, at its best, a numbingly simplistic, static way to represent a 'page' of information, let alone a hypertext system. At its worst, it is a nightmarish wrong turn, given the sheer amount of HTML-based information online today.[4]

This is all detail, though. The more significant impact of the Web as all-encompassing information technology paradigm is that it has drowned out all comers. The enormous promise of new media has been realized—for the time being, anyway—as the business of making Web pages, of gathering Web audiences, holding them by hook or by crook, and the stultifying instrumentalism of marketing logic. The black box of new media has been closed, for now.

Translation #8: From Closed to Open Systems

If there is an antidote to the unquickening of new media it is the mass growth of *open systems*: that the architecture of new media is based on openly published specifications and

4. HTML's historical antecedents—Engelbart's NLS system (circa 1968); Kay's Dynabook vision from the 1970s, van Dam's Intermedia system from the 1980s, to name a few—all go significantly beyond the Web's capabilities. Despite current efforts such as the "Semantic Web" (Berners-Lee et al. 2001) and discussions around a "Web 2.0" which emphasize two-way communications and something closer to a direct-manipulation interface for the Web, the *vast* majority of the ten billion or so pages that Google currently claims to index are static, unstructured HTML files; a truly unfortunate circumstance, given Engelbart and contemporaries' research on information structure, Kay's work on distributed message-passing objects, and so on.

standards. This is in fact the heritage of the Internet going back to ARPA days, and to a very real extent, it underlies the continued success of the Internet. The creation of the Internet as an open architecture, and the fact that this "end-to-end" architecture was conceived as *and remains* application-neutral (Saltzer et al. 1984) is an immense and far-reaching achievement, possibly on a par with the greatest public works projects of history.[5] The Web itself, for all its faults, has at least proceeded according to similar processes; the architectures and standards that make up the Web are open; no one owns them (despite a few attempts), and the Web has come to be a kind of *de facto* mass publishing medium. The jewel in this crown, in my eyes at least, is the distribution of *Free and Open Source Software*; that is, software which is developed, distributed, and used without significant financial or marketing infrastructure in place.

Free and Open Source Software (FOSS) represents a movement that could only have happened with an Internet, and probably couldn't have flourished without something as ubiquitous as the Web. The open and widely available communications (and organizational) medium of the Web seems to have allowed the widespread distribution not just of software but of software *development* across thousands of geographically distributed individuals; what previously could only have been done by people working in close proximity (and therefore requiring something like a capital base) was now possible on a distributed, and in a sense casual, basis. And, while the content of this movement is, self-reinforcingly, the very software that underlies the Internet and Web, the amazing thing is that out of this matrix has come a large-scale, non-commercial model for software development and dissemination. Developer-*cum*-'ethnographer' Eric S. Raymond wrote influentially of the "Cathedral and the Bazaar" as two approaches to software development. The former is the model used in industry and academic research, in which an elite class labours in isolation, periodically handing down the fruits of its efforts. The latter, which characterizes the open source software community, resembles "a great babbling bazaar of differing agendas and

5. The "information superhighway" metaphor directly recollects the American interstate highway projects of the post-war period, a public infrastructure program which clearly shaped an entire nation.

approaches," (Raymond 1999*a*) where a great deal can be accomplished by many hands working more or less together.

Ironically enough (or perhaps not quite enough), the resulting "black boxes" here are rather "open boxes" in that the ideals of the free and open-source software movement exalt the ability to go to any piece of software and dig into it, to look under the hood, and to modify it as required. Of course, as a cultural movement, the FOSS movement has its own share of black boxes; its sacred cows are no less sacred for being "open." But in this ethic of free sharing, participatory development, and emphasis on personal empowerment we are witnessing something very different from the commodity PC industry of the past two or three decades.

The Web as an Educational Medium

Translation #9: From learning experiences to an economy of learning objects

With the advent of cheaply available Internet and the growth of the Web as a publishing medium in the mid 1990s, educators got very excited. For the reasons outlined above, the Web presented a sort of universal multimedia platform, and the ability to access an enormous variety of resources *very inexpensively*. The obvious result was its widespread adoption in educational settings. In a sense, the Web as information resource was the antidote to the packaged curriculum (CD-ROM) trend of the early 1990s. Since the software for browsing the Web is essentially free and the technology and skills required to use it are widespread, the costs of using the Web are limited to the costs of hardware and connectivity, making it an appealling choice for teachers and administrators with limited technology funds. The popular reputation of the Web as a universal library or as access to the world's knowledge has led to the romantic rhetoric of children reaching 'beyond the classroom walls' to tap directly into rich information sources, to communicate directly with scientists and experts, and to expand their horizons to a global perspective.

In this model, *access* is the prime mover; technology equals access. We use technology to get to information, or we're busy making information available to others. Under the rubric of access, "breaking down the walls of the classroom" (Hiltz and Turoff 2000), increasing choice for students or parents, "any time, any place learning" (Harasim 1990), content repurposing, the promise of integrated learning environments—part CAI-style drill-and-practice, part surveillance system (Boshier and Wilson 1998)—media convergence, and good old novelty, we have in recent years witnessed the blossoming of a substantial "e-Learning" industry. E-Learning is largely about the logistics and management infrastructure of education: about vendors, service providers, standards and standards bodies, accountability. The *Instructional Management Systems Global Learning Consortium*, a vast alliance of publishers, technology companies, and educational institutions, aims to provide a set of standards for the exchange and integration of all sorts of e-Learning components and services (IMS Project, n.d.). William H. Graves of eduPrise.com writes of the "ultimate goal of facilitating the acquisition of component parts from a range of suppliers in the educational value chain of nonprofit and commercial interests" (Graves 1999). The language here alone speaks volumes.

Media historian David Noble, in his "Digital Diploma Mills" (1999) lays bare the underlying structure and logic of the e-Learning industry, much to the chagrin of its participants and boosters (e.g., White 1999). Noble points out that the boom of correspondence schools in the 1920s is being reprised today, with similar implications: a blurring of private and public institutions and offerings, a shift toward the commodification of learning materials, and economic implications such as the trend toward increasing enrollment (as opposed to completion and accreditation) as an end in itself.

Popular e-Learning buzzwords reveal aspects of the industrial and commodity-oriented nature of online education: *Course Management Systems* (CMS); *Learning Management Systems* (LMS); *Managed Learning Environments* (MLE). Without even stopping to unpack these Taylor-esque names, we learn that such systems typically do two things: first, they provide an environment in which a 'course author'—sometimes this is a

teacher or professor, but not necessarily[6]—can assemble various pieces of 'content' into a curriculum and add various online tools and resources: discussion spaces, shared filespace, quiz 'engines,' and the like (see WebCT, the industry leader). *Courseware* as such blurs the author/publisher role somewhat, in that it aspires to make a universe of "learning objects" (Henderson 1999) available to a "course author" for orchestration and presentation in a given e-Learning context. The second thing that an LMS commonly does is provide tools for an instructor or administrator—in better cases, an individual learner—to 'manage' individual learning experiences with the courseware: by keeping track of which "learning objects" have been accessed, which tests and quizzes are appropriate when, and what grades are at any given point. A common ideal here is to allow the learner to 'personalize' her learning environment, or at least to customize the visual interface to it. What is seriously at issue, however, is the extent to which the learner is in fact 'managing' her own learning vs. the LMS 'managing' the learner. An LMS trades in standardized educational components—the "learning objects"—and clearly, the ideal for LMSes is to be able to participate in the free trade of learning objects from a wide variety of sources (IMS Project); a sort of NAFTA for lesson plans. Apple Computer's initial (late 1990s) foray into this arena was un-ironically named the "Educational Object Economy"—a standards-based clearinghouse for educational Java applets. So, the resulting black boxes of this translation are commoditized educational resources, educational standards, and the attendant level shift: away from any individual's experience to the semiosis of networked components. *How far we have come from the Dynabook!* Kay's words indeed seem quaint now:

> The particular aim of LRG was to find the equivalent of writing—that is, learning and thinking by doing in a medium—our new "pocket universe." (1996*a*)

The personal computer has been enlisted as the means to access the managed and packaged world of the learning object economy. The only thing *personal* about it is the ability to set preferences, change 'skins,' set bookmarks, post comments. Kay's "curriculum of user inter-

6. See Bryson 2004 for a case study of some of the implications for traditional policies of academic freedom in the face of commoditization of learning experiences.

face" in which one's interactions with the computer were to follow a personal, exploratory, constructive path has been reduced to a stock and standardized menu of choices, in which the only exploratory and constructive options concern which link one clicks on next. Even from the 'authoring' side, those of us involved in the design and creation of online resources and environments (educational or otherwise) are hemmed in by all-encompassing UI standards and the imperative to make things 'Googleable' by composing appropriate metadata and—most importantly—branding things with unique CamelBackedNeologisms to ensure that they survive the sea of search results. Learning has indeed become "enterprised up" (Haraway 1997, p. 70).

The Dynabook Today: How Far Have We Come?

What, then, is *personal computing* in the early years of the 21st century? What is its relationship to education, defined broadly?

I take myself and my own practices as illustrative (if not typical) here. I sit, working through the daily management of texts and tasks and ideas and responsibilities, in front of a laptop computer. This machine is mine; I do not share it with anyone else; its hard drive is full of the accumulated debris of several years' worth of material—*documents*[7] of one kind or another—that I have collected and/or produced. The machine is connected to the Internet most of the time, though I carry it around to various physical locations, and as such it acts as my own personal interface to the global network. It is, in this sense, a typical "personal computer" of this day and age.

My day-to-day practices in this laptop-mediated environment bear the traces of the past three decades of computing. The operating system on my circa-2003 PowerBook is a merging of 1970s Unix and 1980s Macintosh metaphors and systems; its software tools too

7. That our current computing landscape is dominated by "documents" is ironic given Xerox's ("the document company") ambivalent involvement. The docu-centric division between *applications* and *files* can be traced to the Unix software culture in ascendancy in the 1970s and which had a impact on the early microcomputer market of the1980s. It is actually in sharp contrast with Kay's Dynabook vision, in which media objects featured both contents and computing intelligence. The "file" metaphor, when combined with the "authoring" software of the early 1980s—word processors, spreadsheets, paint programs—becomes translated to the arguably friendlier "document" metaphor, still ontologically distinct from "applications."

are a blend of Unix-derived command-line programs; graphical, single-user Mac-style "authoring" programs; and Internet clients (electronic mail, web browser, file-sharing tools). These three realms of software rarely overlap, and to fully use a system such as this is to shift between the cultural traditions these tools represent; I am in significantly different cultural space when I am using e-mail (largely defined circa 1970), writing in a word-processor (circa 1985), manipulating a photograph in Photoshop (circa 1992), or searching the Web via Google (circa 2004).

In terms of day-to-day productivity (defined as much by the shape and practices of the contemporary workplace as by software), Internet-based personal communications and reference-browsing makes up the bulk of my actual computing practice. After this comes document creation; in my case, this is almost entirely about writing and the production of papers and reports for various purposes and audiences, for which I employ a small arsenal of writing and publishing tools, perhaps somewhat atypically, since I eschew the ubiquitous Microsoft Word on political grounds (more about that later). The practices which make up this work (drafting, revising, opening, saving, cutting, pasting, printing) are essentially those established twenty-five years ago when word processing became a black box closely equated with personal computing. The third major aspect of my actual personal computing is one I share with an increasingly large population, and especially those younger than me: listening to music. I have, for a few years now, been listening to recorded music primarily via the computer rather than CD player or cassette-tape player, and my collection of digital music files (which take up *nearly half* my laptop's hard disk) is with me wherever I am. With "rings on my fingers and bells on my toes," as the nursery rhyme goes, I will have music wherever I go. Digital files—MP3s being the ubiquitous format at this point in history—are such an improvement in convenience over physical formats like discs or cassettes that one might think that this is another "killer app"[8]— one that the recording industry is notoriously having a hard time dealing with. A similar trend to keeping and playing music as digital files

8. As evidenced by the iPod personal music players this trend is not lost on Apple Computer, though it is yet unclear what the relationship between iPods and personal computing is.

is digital photography, and, ever since my kids were born, my laptop has become my photo album as well.

A number of observations on this little portrait of personal computing are in order. First, and notably, there is very little "computation" going on in any of what I have just described; the operative model is much more one of a small set of data formats (e-mail, web pages, written documents, MP3s, photos) and a software toolset for managing and sharing them. Second, the ubiquity of the Internet makes my personal computing an extension of—or perhaps a replacement for—public and private communications systems like the telephone, television, and print publishing. Third, with the exception of writing and taking photos—practices at least theoretically independent of the computer—there is almost no "authoring" or creative expression going on here at all; rather, these tasks are either formally communicative (i.e., the production of highly generic forms like documents) or relatively passive. I am inscribed as either a commentator or a consumer of digital media; despite the powerful tools at my fingertips, very little of my day-to-day computing involves creativity or even exploration.

It is instructive to examine this portrait in the light of the Dynabook vision. Some of what Kay had in mind in the early 1970s is clearly present: the basic form factor and hardware capabilites of todays laptops are very close to what Kay foresaw: a truly portable device, connected wirelessly to the global information utility, and capable of presenting and integrating a variety of different media (text, image, audio, video). But, seen a slightly different way, the personal computer of 2005 is more akin to a TV set than a computational medium. In 1972, Kay wrote:

> What then is a personal computer? One would hope that it would be both a medium for containing and expressing arbitrary symbolic notations, and also a collection of useful tools for manipulating these structures, with ways to add new tools to the repertoire. (p.3)

I have to admit that the ways in which I "manipulate arbitrary symbolic notations" on my personal computer are *few and far between*. Word processing is the one notable exception;

almost every other operation on digital information is one of presentation or filing (sorting, archiving, searching). Adding "new tools to the repertoire" is also underwhelming; this can only refer to purchasing or downloading new application software.[9] I have on my hard drive software tools for creating and manipulating images, composing and arranging electronic music, editing video—each application living in its own narrowly constrained domain, poorly (if at all) integrated with other applications, even the dominant ones of e-mail, web browsing, word processing. Further, there is something troubling about the "arbitrariness" of these "symbolic notations;" in practice, they are not very arbitrary at all. Rather, they are established, *standardized* notations: one for text documents, another for bitmapped graphics, another for electronic music, and so on. In a *data-centric* world, standards for data representation (file formats, etc) are essential to make content exchange possible. Note that this is in contrast to Kay's vision of a world of message-passing objects with dynamic and *negotiated* context and semantics. If our symbolic notations had the kind of "arbitrary" character Kay had in mind, we (users) would perhaps come up with new ones now and then, even on a personal level; we might blend them, modify them, experiment with them. But to even muse on such possibilities in our workaday world puts us in the land of either idle speculation or marginalized geekdom.

If my personal portrait of personal computing is underwhelming, the mainstream reality of educational computing is even more so. Educational computing must at this point be taken as a subset of personal computing—the application or recontextualization of the sorts of application and communication software I have been describing to classroom and curriculum use—and, unfortunately, often with the "personal" qualities removed or limited for adminstrative reasons: shared computer labs which necessarily prohibit even the sort of messy *habitation* of a personal computer that I have described here; top-down adminstration which puts severe restrictions on the use of programs and the storing of personal files;

9. Despite my relative comfort with a half-dozen programming languages and even some substantial experience with software development, this does not fall within the realm of my "personal computing." Nearly all the software I have written myself has been for someone else, and within the client-server model of the Web. The one exception is a personal bibliography-management tool I created and still use—an anomaly in the model I have been describing.

pre-packaged and pre-defined possibilities. Remember that in Kay's vision, personal computing *began* with the educational context, and *out of this* would come the day-to-day personal computing of adult professionals. Educational computing would have been the superset, the productivity tools used by adults a specialized subset of the possibilities explored by kids. We have instead got the reverse.

Vendorcentrism

One major factor Kay may not have imagined is the role that key software and hardware companies play in mediating our personal computing experiences. If we believe the common statistic that only about 1% of desktop computers use the open-source Linux operating system, we can surmise that for 99% of us, personal computing is something we do via a toolkit created and sold by a large American corporation, and in the vast majority of cases, it is Microsoft.

It is not my intent here to analyze the dynamics of a market-driven computing landscape; such a study would constitute a separate project, one at least as large as this one. But it is not much of a stretch to suggest that a truly malleable personal computing environment of the sort Kay envisioned might be very difficult for a commercial operation to market; a far safer and more established model is to offer consumers a relatively small palette of choices, and to carefully and proactively manage their expectations. Mergers of IT companies with media and entertainment firms (e.g., AOL/Time-Warner) seem to strengthen this general trend. Even without speculating on the details of corporate motives and strategies, it is clear that personal computing has since its very beginning been dominated by a very small number of companies weilding enormous power. The market dominance enjoyed by Microsoft is in practice a *single point of interpretation*, or in Latour's language, an *obligatory passage point*; a single entity positioned so as to exercise unparalled control over what personal computing means, what it includes, and what is possible within its horizons. Our practices with respect to IT are necessarily defined by this agency. That there exist other large computer companies does little to offset this: the contributions of the 'second string'

of corporations—Apple Computer, Sun Microsystems, IBM, AOL/Time-Warner—are in most cases merely lesser versions of the Microsoft model.

"New Media" vs. "Cyberculture" in the 21st century

The opposition between "new media" and "cyberculture," proposed by Lev Manovich in his introduction to MIT Press' *The New Media Reader* (2003), is part of Manovich's effort to define "new media" by elaborating a historical perspective (informed largely by art history) on digital technology and its relationship to culture and cultural production. Manovich offers eight partial definitions of new media, but the first one—that it is distinct from what he calls "cyberculture"—is especially food for thought:

> In my view, [new media and cyberculture] represent two distinct fields of research. I would define cyberculture as the study of various social phenomena associated with the Internet and other new forms of network communication. Examples of what falls under cyberculture are online communities, online multi-player gaming, the issue of online identity, the sociology and the ethnography of e-mail usage, cellphone usage in various communities, the issues of gender and ethnicity in Internet usage, and so on. Notice that the emphasis is on the *social* phenomena; cyberculture does not directly deal with the cultural objects enabled by network communications technologies. The study of these objects is the domain of new media. In addition, new media is concerned with cultural objects and paradigms enabled by all forms of computing and not just by networking. To summarize: cyberculture is focused on the social and on networking; new media is focused on the cultural and computing. (p. 16)

In reflecting on this distinction, it occurs to me that the vast majority of academic and popular/journalistic discourse around computing (personal or otherwise) in the past decade has not been in the "new media" space, but in the sociological realm of Manovich's "cyberculture." Furthermore, it is largely the attitude toward the division of labour between experts and end-users that leads me to this identification; we talk of the "effects" of computerization, of "social impacts," of "user friendliness," of "no programming experience required," or of "making the technology serve pedagogical ends first"—clichés which inscribe in the first place to a divide between technology and society (cf. Latour 1993), and in the second place

further reify the division of labour between experts and users; assumed is a class of quasi-magical designer/engineer types, who are somehow not quite human, (yet) who wield enormous power, handing technologies down to the rest of us ('hand-me-downs from the military industrial complex,' according to one popular notion), who then are forced to use these inhuman and dehumanizing tools which never quite meet our predefined aims and goals. Inscribed here too are the circumscribed possibilities of a so-called *resistance* in which the content of cyberculture is the impotent critique of its own form—impotent because it is blind to and thus further reifies its own political-economic conditions; that is, the *market* as the inescapable model for all discourse.

This characterization is of course the market's own self-perpetuating reality; generated and sustained because it makes good business sense to do so, as has been proven time and time again in examples such as Microsoft's and Bill Gates' surreal fortunes. Now, this situation would be servicable if it ended there, and the now almost traditional themes of resistance and culture-jamming and the romantic ideal of the arts could be called upon to prevail over the forces of oppression. The frightening part, however, is that as digital technology becomes more ubiquitous, our collective implication in this divide seems to become deeper. As the World-Wide Web became a daily tool and information source for larger and larger segments of the Western world in the late 1990s, the sense of collective helplessness seemed to be even more entrenched, despite the early talk of democratization that accompanied it (e.g. Landow 1992). This notion of democratization may have been simplistic, but it was not without merit; I recall, in 1993, after being paid to create web pages for the first time, remarking that no one would ever make any money this way, that it was so simple that trained monkeys would soon be doing it; and yet, by the time the *dot-com* boom was in full effect in the late 1990s, budgets for websites reached to hundreds of thousands and even millions of dollars, and a new 'professionalism' turned the Internet—potentially the most participatory institution in history—into something beyond the curtain, more and more like television in most people's lives.

To ride Manovich's framing a little longer, it indeed appears that "cyberculture" has been in ascendance for the past decade or so, and "new media" has remained marginal; this parallels the difference apparent between my portrait of contemporary personal computing and what Alan Kay might have had in mind. The implications of such a cultural trend for education are profound: given its status as a spectatorial, market-dominated discursive space, what does "cyberculture" offer education, exactly? How do we ensure that it does not become the IT equivalent of "music appreciation?" This critique is, I believe, a microcosm of the much larger criticism made by Homi Bhabha of the language of "cultural diversity"—that, in the liberal tradition,

> although there is entertainment and encouragement of cultural diversity, there is always also a corresponding containment of it. A transparent norm is constituted, a norm given by the host society or dominant culture which says that 'these other cultures are fine, but we must be able to locate them within our own grid' (Bhabha 1990).

In the case of cyberculture, the "transparent norm" is provided by the market logic of the industrialized Internet, which easily accommodates critique, diversity, even radicalism, while maintaining and reproducing the 'means of production': hence, cultural diversity and even resistance are articulated within the familiar bounds of Microsoft Word. This little example is facile, trivial, in comparison to the analysis of discourse and truth effects that Foucault mounted three decades ago. Do we see anyone calling this out, or working on alternative channels? In techie circles, yes; but in academia and education, no.

Lessons from the Open-Source Movement

> *Unix, or Linux, is the stone-soup operating system, where everybody has brought their own contribution to the common pot.*
>
> – Tim O'Reilly, 2002

There is, thankfully, an alternative to the pervasive corporate-dominated computer industry, a growing force in the past decade. The truly interesting developments in computing in

the last decade have almost entirely taken the form of radical innovations from the fringes which is then transformed by some large existing corporation or other mass of capital (more fluid in the dot-com boom of the late 1990s) into marketable form; much of the Internet's translation into a so-called 'public sphere' has been a shift on this level. But that a marginal realm *even exists* from which innovations can emerge and where interpretations remain somewhat more fluid is an encouraging thing. The relative importance of this margin has been shored up in recent years, as well, in the reification and institutionalization of the Free/Open-Source movement.

The Free and Open Source Software (FOSS)[10] movement came to popular attention in 1997 when the rhetorical implications of the words "free" vs "open" were problematized, bringing a long-standing but increasingly marginalized *tradition* to a point of encounter with the corporate world. This is a tradition of software developers sharing their creations freely with one another, a practice which, in certain sectors of the IT world pre-dates any corporate or market involvement, and which became formalized in the mid-1980s by Richard Stallman (1985; 1998), founder of the Free Software Foundation, in response to what he saw as an erosion of the collaborative community of programmers he had come to know in the 1970s and 1980s.

Stallman's articulation of the rationale and ethics of sharing software put a formal face on a practice that had been widespread, though informal, for decades. The ARPA community in the 1960s widely shared the fruits of their efforts, seeing their work as an extension of the scientific/academic tradition of publishing research so that other researchers could build upon it. The underlying technologies of the Internet and most of the computing architecture surrounding it were developed and disseminated according to these ideals. The *Unix* operating system and the community and tradition surrounding it (since the early 1970s) most clearly embodied this model of development, since it meant that the development community was distributed, consisting of programmers and researchers from a

10. The rather awkward moniker "Free and Open Source Software" attempts to be inclusive of both the more radical "free" idea and the business-friendly "open" term.

number of different sites and institutions, some corporate and some academic. Unix arguably became the dominant computing platform of the 1980s—and ultimately the key platform for the Internet—because of this development and distribution ethic.[11]

But what Stallman saw in the early 1980s was a "stark moral choice" (Stallman 1998) presented by corporations increasingly interested in protecting their 'intellectual property'—Stallman believed that he must either take action to counter the trend toward corporatization of software, of find another career. His efforts, as rationalized in the *GNU Manifesto* (1985), culminated in two important contributions: the beginnings of a GNU operating system—a formally free version of Unix, unencumbered by typical corporate licenses[12]—and the GNU General Public License (GPL), which is written so as to do the opposite of what most software licenses do. Instead of focusing on the owner's control of the software and granting rights of use to licensees, the GPL ensures that the software's source code (that which a programmer writes or modifies) remains open and freely available, no matter who (or what corporate entity) contributes to it. It is a copyright document—as Stallman put it, a *copyleft* document—that subverts the traditional copyright concept, ensuring the freedom of users rather than restricting use. The GPL's secret weapon is not that it merely says you can do whatever you want with this software, but that it stipulates that anything you create based on GPL-licensed software *also* falls within the license—that is, the source code must always remain open. This has the effect of preserving and propagating the idea far beyond its original application. The GPL establishes a specially formulated *commons*; software licensed under the GPL cannot be restricted, in use or distribution; it guarantees that the source code remain free for anyone to access, use, or modify for their own ends.[13]

---

11. On the Unix tradition, see Eric Raymond's *The Art of Unix Programming* (2003)—a work of cultural ethnography at least as much as a guide to software development, which proceeds by tracing the virtues and values emergent within this now venerable culture of computing.

12. GNU is a recursive acronym for *GNU's Not Unix*. The project was to produce 'copyleft'-licensed components making up an entire, usable operating system based on Unix. Unix had roots in academic and sharing-friendly environments, but was actually owned by AT&T and licensed to various various competitive vendors. Stallman's GNU project made steady progress through the late 1980s, but the real breakthrough came in 1991, when Finnish programmer Linus Torvalds released a free Unix-like system kernel under the GPL license. Torvalds' "Linux" kernel effectively completed Stallman's GNU project, making it possible to download and install a completely free Unix-like operating system for (by then ubiquitous) Intel-based PCs.

13. The GNU General Public License (GPL) can be found at http://www.gnu.org/copyleft/gpl.html

Through the late 1980s, as the Internet was growing in size and profile, the GPL was applied to a number of key Internet infrastructure components, the result being that the Internet was largely constructed out of free software, and the tradition of free software was firmly established within the early Internet community. By 1997, GPL-licensed Internet software—and the GNU/Linux operating system—had risen to sufficient popularity that Microsoft began to comment on the threat that this development movement presented to its market dominance. Eric Raymond's strategic rebranding of the free software movement as the "open source" movement capitalized on this popularity. Raymond felt that Stallman's rhetoric, with its stark moral terms and talk of "free software" might alienate corporate America and therefore undermine the popularity of software written and released under such terms (1999). Raymond's efforts seem to have been a success, at least in terms of raising the public profile of Linux and other "open source" projects.

Historical details aside, the importance of the FOSS movement—which remains largely centered around the Linux operating system—isn't in the details of the license terms, or in the specific vocabulary used when pitching it to corporate audiences. Rather, it is in the strength and coherence of the community which has emerged around it. The FOSS movement now commands enormous "mindshare" if not marketshare (we might alternatively say that it has great discursive influence). While remaining a very small player in the desktop-based personal computing environment (accounting for somewhere between 0.5% and 3%, if we believe statistics as reported in *The Economist*, April 15, 2004[14]), Linux and FOSS' share of the institutional and web server market is *much* greater: Linux accounting for between 1/4 and 1/3, according to most sources, with particular open-source applications (e.g., the Apache webserver) boasting even higher marketshare. Moreover, the community surrounding these projects and their deployment have come to self-identify and rally around a few key ideas:

1. the promethean ideal of a viable alternative to Microsoft's near-monopoly;

14. My point here is not that *The Economist* should be distrusted on this issue, but rather that it is very difficult to gain meaningful statistics on how many copies of a particular free software system are running. How would one count them? There are various strategies, but compared, say, to counting how many Windows licenses are sold, it remains very inexact.

2. the ideological stance about corporate ownership vs freedom ("as in speech, not as in beer") as proposed by the GNU Project's Richard Stallman;
3. the notion that "open-source" development strategies lead to higher-quality sofware, as promoted by Eric Raymond;
4. the do-it-yourself example (and success) of community leaders like Linus Torvalds;
5. the economic fact that FOSS systems of very high quality can be downloaded, installed, and used for 'free.'

The fact that some of these ideas and ideals may be in tension or conflict has been analyzed extensively in FOSS-oriented online fora and in scholarly journals (especially *First Monday*), but the coherence of the open-source community is readily apparent. In what might be characterized as one of the central organs of the Linux community, the *Slashdot* news forum, the self-righteousness of the FOSS model is so often proclaimed, and deviations from the party line so roundly denounced, that the overall effect is sometimes something like a political rally. Furthermore, the amount of ongoing attention that FOSS receives in such mainstream magazines as *Forbes* and *The Economist* serves to show the impact it has on IT culture.

Of Unix and other IT cultures

To speak of IT *culture* at all is to step out on a limb; there has been little if any ethnography to document the rituals, kinship structures, and worldviews of various computing-centric communities. But if, as postcolonial critic Homi Bhabha suggests (1994, p. 36) culture is knowable only in the experience of *difference*, then the moment of encounter with something like Linux puts it in sharp relief: something very *different* is going on here than in the mainstream PC market. Alternatively, if we treat—as I am wont to do—culture as *history*, then it is fair to say that Linux and the FOSS projects surrounding it are extensions of the much more venerable Unix tradition. Unix was originally developed by hackers at AT&T Bell Labs in the early 1970s and spread almost despite any formal efforts through the 1970s

and early 1980s by individual sharing; AT&T had been judged a monopoly by the US Justice Department in the 1950s, and the regulations which followed this ruling effectively prevented AT&T from bringing software to market directly. The popularity of Unix in the 1980s led to its being controlled by a number of licensees who marketed and protected it aggressively; this in part was the situation Stallman felt he must react to. Unix' role in the development of Internet standards, as well as the emergence of alternative Unix software under the GNU project and then Linux, set the stage for something of a renaissance in the 1990s.[15]

To know Unix (and/or Linux) well is to come to appreciate its history; this is much more important than in any other computer system in widespread use. Though most Unix and Unix-like systems today have familiar-looking windows-and menus graphical interfaces, the "Unix way" is to type interactively in a command shell (the 1960s time-sharing paradigm lives on); the commands one issues are in many cases only understandable by reference to the historical contexts that produces them (see Raymond 2003 for a full exposition of this theme). When one encounters the Unix command *tar*—useful when packaging up files for backup or distribution—it is only when one realizes that was originally shorthand for "tape archive" that it begins to make real sense, despite its continued existence and daily use on systems with no tapes anywhere near them.

Novelist Neal Stephenson wrote that "Unix is not so much a product as it is a painstakingly compiled oral history of the hacker subculture" (Stephenson 1999). A better characterization—and one that avoids the clichéd, pith-helmet anthropologist trope—comes from software philosopher Ward Cunningham, who observed that "Unix is the Latin of computer science," to which some wag replied, "Except it ain't dead yet." (WikiWikiWeb: UnixCulture).

Not dead by a long shot. *Unix-as-Latin* is a pretty good analogy (far more apt than Koschmann's "Logo-as-Latin"). We must bear in mind, though, that this would be Latin

15. I am, for simplicity's sake, completely ignoring the role of the Berkeley Standard Distribution (BSD) of Unix, and its three main open-source versions. BSD Unix is technically closer to the pure Unix tradition than Linux, but the more popular Linux has become symbolic of the FOSS movement.

circa 1700 or so, not Latin as it exists today, for the simple reason that a large proportion of the 'scholarly' class still speak it (Unix), and one cannot truly enter that class without knowing at least a little of it. It is difficult, obscurantist, and exclusive. It is also brilliant, elegant, and—judging from Linux and FOSS—currently in its *renaissance*. Its impact is huge and wide ranging. When one stops to consider that Unix and its direct descendants (like Linux) are the fruits of Licklider's ARPA vision (almost literally kept alive through the dark ages by monks) and that they serve as the foundation stones of the Internet and the Web; that the FOSS movement has made such important waves in the IT industry (even Microsoft talks about it); and that every contemporary operating system inherits something from it (Apple's 'new' OS X is a GUI on top of a Unix foundation layer, and even Microsoft's Windows draws from Unix on several counts); one begins to appreciate its pervasiveness, and—more importantly—its resilience in the face of wave after commercial wave of 'innovation' designed to make it obsolete. Nikolai Bezroukov, noted for his cutting critiques of Eric Raymond's open-source anthropology in *FirstMonday*, noted that "UNIX Renaissance OSS proved to be the most *important democratic movement* in software development in the 20th century" (Bezroukov 1999).

Latin it may be, but herein lies the problem with the Unix/Linux/OSS tradition as a would-be educational model: *the analogy with Latin is a little too apt*. The Unix tradition, like that of Latin in the Renaissance, is frankly stubborn, dogmatic, obscurantist, sexist, and exclusive. It rewards those who *can*, whether by sheer dint of will or by having access to the resources to struggle their way to the inside of the citadel. Unix culture has little patience with those who cannot or have not, chiding them instead with RTFM—*read the fucking manual*—the scathing words of the Unix culture-hero, the "bastard operator from hell." In the FOSS world, the admonition is the politer-sounding but ultimately more daunting "read the source code." The gathering places of Unix and FOSS culture, like the *Slashdot* forum ("News for Nerds"), are rife with arrogance, conceit, and condescension; these can be unfriendly places for the uninitiated.[16]

That said, there is no doubt that the tradition which comprised Unix and the bulk of the FOSS movement is a *curriculum* par excellence, that it constitutes a *literacy* in the richest sense of the word. It is a vast and multi-layered community of practice, precisely as Lave and Wenger (1991) would have it (for this angle, see Tuomi 2000; Hemetsberger & Reinhardt 2004), and it is arguably a more efficient learning environment than many schools provide, judging by the sheer scale and growth of the movement. But it is not, I argue, the kind of learning environment anyone would purposefully design, nor *choose*, if not for the simple dearth of technological alternatives. What is the lesson in that gap? Despite the growth in popularity of open-source tools as educational environments, and a semi-organized community of "open-source in education" advocates on the Internet, these advocates' approach seems to be practical and incremental rather than visionary.[17] Unix culture is, ironically, *a highly effective curriculum built on absolutely perverse pedagogical principles*—chaotic at best, and positively hostile in many cases. What lessons can we draw from that?

My intent here is not to pass judgement on Unix or FOSS as they stand today, given my general admiration of them and hope for their possibilities, and given the likelihood that this tradition will continue to flourish and evolve. Rather, I mean to position the Unix/FOSS tradition as a major historical actor (or actor-network), and to point out some of the enormously generative spaces it has opened up in the past decade: a *real* democratization of software development, a non-commercial development and distribution model, a huge community of practice operating with virtually no formal institutional support, and an evolving ideological frame (exemplified by Stallman's *copyleft*) that is being energetically adapted to other realms: publishing, file sharing, and so on. And finally, I mean to point out, as Alan Kay has, that there seems to be an interesting 30-year lag between the time a powerful idea is first worked out and when it reaches the mainstream.

---

16. To be up front about my positionality here, I make no claim to being "uninitiated" in this culture. I have been running Linux- and Unix-based systems at home and at work since about 1997 and have participated in a number of small-scale open-source development projects. That said, the exclusive atmosphere of the community is still readily apparent to me, despite its democratic rhetoric.
17. See "Moodle" (http://moodle.org/), an open-source course management system that is gaining widespread popularity among educational institutions. See also the "Open Source in Education Foundation" (http://www.osef.org/)

To bring us back to the Dynabook theme, recall that Unix's initial incarnation precedes the first Smalltalk by only a few years, that these two technological threads are close historical siblings.[18] *What if we were presented with an equally venerable model of computing that featured many of the good ideas inherent in the Unix/FOSS culture—openness and sharing, simplicity and modularity of design, collaborative and network-based architecture—but one which was designed with pedagogical ends in mind?*

Authoring in the Age of the Web

The great, visible, and lasting legacy of Unix culture is, more-or-less, the Internet and the World-Wide Web. Since some time in the mid 1990s, the Web has dominated personal computing to such an extent that it has in some ways become invisible: Tim O'Reilly (1999) noted that the Web itself wasn't the "killer app" that made people want to buy a computer in the 1990, *Amazon.com* was. The ubiquity of the Web, both in terms of its physical manifestation and its sociocultural embeddedness have made it a background field, the invisible infrastructure (Star 1999) that provides the ground upon which our more focused deliberations appear. That bedrock sensibility points to some tightly shut black boxes, and it is here that I want to draw our attention.

Because the Internet, the Web, and the FOSS movement all draw on Unix culture, the conceptual motifs—genres—of the Internet and Web of the 1990s and early 2000s draw heavily on the Unix worldview; the architecture of much of the existing Internet and Web, as a result, is Unix-like. It is not surprising, given this, that Unix has experienced a renaissance in the age of the Web. Conversely, the community dynamics that surround projects like Linux are dependent on the existence of an Internet and on technologies like the Web. Indeed it is unlikely that projects of the scale of Linux could have flourished without the kind of sociotechnical foundations provided by the Internet and Web: mailing list commu-

18. John Unsworth (1995) notes the apparent paradox in the 'open,' 'public,' even 'democratic' technologies springing from the "pure research" contexts of American monopoly capital—the ARPA project, AT&T, Xerox—and that their relative insulation from quarterly results and other "real world" economic pressures that such collossal institutions could uniquely provide gave rise to technologies of unparalleled scope, while enshrining at a deep level the structures and assumptions of liberal capital: individualism, property, labour, and power.

nities; networked hypertext documentation and file repositories, Internet-based version-control systems, and the kind of free-flow of information that Unix culture has always relied upon.

The flip side of this, which is most relevant to our consideration of the Dynabook, is that the Internet—as a vast network of Unix and Unix-derived systems—is significantly *not* a vast network of message-passing Smalltalk objects. And though Unix and Smalltalk projects are both products of the early 1970s, and although they share many virtues in common—openness, modularity, simplicity, to name a few—they are distinct 'paradigms.'[19] Where Unix is firmly grounded in the terminals-and-mainframe timesharing model of the ARPA project, and its programming paradigm (enshrined in the language C, a co-development of the Unix project) is one of *algorithms operating on structured data* residing in files (this model so dominant and ubiquitous today that it escapes analysis for the most part—see Lanier 2006), Alan Kay's vision of Smalltalk thirty-five years ago was a reaction to and turning away from this very model, away from the data-vs.-procedures distinction which underlies it. Kay had in mind *personal* computing objects which interacted in a peer-to-peer fashion; instead of distributed access to centralized computing resources (the ARPA vision, which Unix implements), the Smalltalk vision is one of a *radically decentralized* network, in which each personal node is as powerful as any other, and where computing can occur in any combination, local or distributed).

19. I use the word 'paradigms' here advisedly, and in Kuhn's (1996) sense, I mean that beyond a superficial level, the virtues of Unix culture seem distant from a strong OOP/Smalltalk perspective, and conversely, that the virtues of object-orientation seem foreign to a 'native' Unix perspective (e.g. Raymond 2003, pp. 126ff).

We have seen fragments of this vision appear in recent years: on the Internet, truly peer-to-peer applications have begun to emerge, often rendered marginal by their questionable legal status (in some circles, *peer-to-peer* has come to be simplistically equated with illegal sharing of music files). And the very ubiquity of the Web and web-based applications has led to a relative downplaying of operating systems *per se*, at least in terms of desktop computing environments.

By and large, however, the centralized model—in more contemporary parlance expressed as "client-server" computing—is dominant, despite the considerable amount of computing horsepower in the average PC. Though truly peer-to-peer computing is more feasible now than it ever has been, consider the following commonplaces: web pages are stored and served from a centralized host (e.g., an academic department, a commercial hosting provider); "web-mail" stores and manages your 'personal' e-mail on a centralized server (though a look at the corporate politics behind Microsoft's *HotMail* service or Google's *gmail* surely raises the question of just how 'personal' these are); Google's search engine indexes the Web centrally. Even apparently distributed applications like blogs are hosted by massive centralized servers. There is absolutely no 'technical' reason why all these activi-

Objects and OOP in the 1990s

I don't mean to suggest that the Unix paradigm is *opposed to* the object-oriented paradigm. There are in fact many points where these ideas can be seen to intersect. While Kay's Smalltalk represents a distinctly different vision from the "Unix way," OOP itself became merged with Unix culture in the 1990s despite skepticism about object-orientation amongst Unix devotees (Raymond 2003, p. 127)

OOP gained enormous popularity and legitimacy in the early 1990s, as books proliferated and languages like C++ began to be integrated in college curriculum. C++ could be integrated into existing Unix-based contexts (both "social and computer-wise," writes Stroustrup) while offering the advantages of classes and encapsulation. Here was an OOP language which existing Unix and C programmers could easily adapt to. In 1995 the introduction of Java was perhaps an even bigger push for object-oriented programming, and it came from Sun Microsystems, a major Unix vendor. C++'s original developer Bjarne Stroustrup writes:

> *In 1995, Java burst upon the programming scene with an unprecedented style of and amount of marketing. This was the first time that the resources of a large corporation had been thrown squarely into a programming language debate. (Stroustrup 1998, p. 290)*

Java's release was wrapped in idealistic, if not revolutionary aspirations. Java took much from Smalltalk, in terms of its virtual-machine architecture and its branding as the "language of the Internet." Java has been a qualified success, especially in encouraging the adoption of OOP concepts in the Unix and Internet world.

But Smalltalk still remains apart; its basic model for interaction and filing are notably distinct from the Unix tradition. In a sense, Smalltalk denigrates the very idea of an operating system, "a collection of things that don't fit into a language. There shouldn't be one" (Ingalls 1981, p. 298). In this sense, Smalltalk is better compared not just to languages like C or C++ or Java but entire operating systems like Unix or Windows.

ties couldn't be carried out on our own 'personal' computers in a peer-to-peer manner, but the way we have been conditioned to think about computing makes the division of labour between *clients* and *servers* seem natural. The very ubiquity of the Web as a client-server application with its standardized user interface strongly reinforces this, and in a sense undermines the 'personal' potential of personal computing. In the very simplest rendering, dealing with the world through a web browser 'dumbs down' our interactions with that world, because it casts every activity into a classical asymmetric relationship: the *brains* of the operation are at the server, the *beauty* is at the client end. It is not surprising that this model has worked out well for marketing purposes. Heaven forbid we be told that we're the brains of the operation.

And, ironically, despite the apparent 'democratization' of Linux and FOSS, these technologies are quite conservative on this architectural axis; they maintain the centrality of the client-server model. Very few people actually use Linux as the client-side of their computing world, and even so, Linux as a desktop environment closely emulates the Windows model and metaphors.

Web authoring and computer literacy

In painting this broad-strokes history, though, it is important to avoid the sense that these stories are smooth and unidirectional. In the case of Unix, FOSS, and the Web, the key point to bear in mind is that this is a story which picks up enormous momentum and relative clarity in the early to mid 1990s. Prior to that—that is, prior to the Web—the threads of this storyline were much harder to discern. If we go back to about 1991 or 1992, we find people working in HyperCard on the Macintosh, which, for all the limitations of the existing software and hardware base of the day, was much more conceptually in line with Kay's vision (significantly, though, it lacked a networking component). Here was "authoring" in an open-ended and personal style, on the personal computer. In a nostalgic piece on the state of graphic and interactive design in the age of the Web, compared with the flowering of a

multitude of design ideas in the late 1980s and early 1990s, designers Groff & Steele write of a "creative golden age" so different from today's web-oriented perspective:

> What strikes us, looking back, is the sheer originality. Work from this period is fascinating because artists and programmers were required to work from a blank slate. The territory was mostly unexplored, so by definition every move required originality. (Groff & Steele 2004)

Similarly, from the standpoint of education and especially IT curriculum, the advent of the Web and "web-page authoring" is in most cases a huge step backwards (or downwards) in comparison with the radical mathematical aspirations of Logo or even the open-ended multimedia authoring of HyperCard. Yet by 1995, both Logo and HyperCard were "history"—despite marginal groups of loyalists, these systems had been all but washed away by the tide of the Web. In the process, the general thrust of "computer literacy" moved away from *authoring and media production* and toward *information access*. Now the dynamics of information and access to it—who has it, who controls it, who provides it—are certainly of critical importance, and it would seem that this is an entirely reasonable direction for educational IT to move in. But before we consider that question answered, let us pause and consider the black box which is being closed here.

The shift away from 'media' and toward 'access' makes an assumption about the underlying infrastructure: that the means and mechanisms are good and sufficient to meet our informational ends, and conversely, that the means and mechanisms are of sufficiently low level and sufficiently uninteresting from an pedagogical perspective that we can safely forget about them, leave them to professionals, and attend to the higher levels of getting to *the content*. The extent to which we willfully forget Marshall McLuhan's cardinal insight here is noteworthy, but this is, I argue, the deal we struck in the 1990s, when we adopted the combination of web browser, HTML, and client-server architecture as our basic means of encounter with the digital world. What we get in return is a world packaged as web pages. Now, considering that Google currently claims to index ten billion web pages, this might not seem like a bad arrangement. But let us continue: what we gave up in the bargain is the

ability to critically engage with the form and structure of that information, which we have left largely (though not entirely) to packagers from Microsoft to AOL/TimeWarner.

Black boxes, of course, are all about such bargains; I contend, however, that from the educational perspective at least, *this box has been closed too soon*. Andy diSessa writes:

> To be direct, information is a shockingly limited form of knowledge. Unfortunately, our common culture seems insensitive to that fact. We even call the computer's influence the "information revolution." There is an information revolution, but it is not the basis for a revolution in education. As turbulent and big as the information revolution has been, I think the truly interesting (and maybe bloody) part of the revolution as regards computational literacy has yet to begin, and society is less well prepared for it. (diSessa 2000, p. 219)

Admittedly, there is more to the Web than web pages *per se*, and beyond that, there is certainly more to the Internet than the Web. And of course the Web in its monotonous Webbiness is also a thing of amazing scope and quality and variety. The point I am trying to make is that the culture of the Internet has its centre of mass in a spectatorial/consumer-oriented milieu, and that most thinking about IT in education has bought into this wholesale. A key example of this is the online "economy" of "learning objects"—this refers to a large scale educational technology trend, in which a set of interoperability standards (e.g., the IMS Project) have been established which enable and encourage the development and exchange of reusable, modular, multimedia educational *content*.[20] The ideal for such learning objects seems to be interactive Java applets (that is, small, Java-based programs that run within a web page) which simulate some complex system or model and allow learners to explore its behaviour. There is no doubt that many such educational applets have been created, exchanged, and indeed re-used in various contexts. But some professional programmer (or, better but far rarer, a teacher) somewhere "authors" such content, while the learner's engagement with the content is restricted to her viewing or playing with it. This is a traditional *one-to-many* publishing model; despite the formidable superstructure

20. Norm Friesen's excellent article, "What are Educational Objects?" (2001) puts in sharp relief the relationship between interoperability and decontextualization in the learning object economy.

of technologies, traditions, standards, and frameworks which provide the learner with these opportunities, it amounts to little more than 'interactive textbooks'—with the control remaining with the publishers and the audience left as an undifferentiated mass. The economics of the publishing (or broadcasting) model are too powerful to ignore: one generates an audience in order to sell access, not to the information, but to the audience.

One cannot deny that the Web is indeed a distributed authoring environment, in that anyone anywhere can put up a web page on their local server. Furthermore, no single agency is truly in control of what content appears. As a result, we have a proliferation of what might be termed "artistic" or "expressive" uses of the web: HTML-based web pages, extensive graphic design work, javascript (scriptable web pages) programming, and visually intense Flash animations and games. It would be dishonest and unfair for me to dismiss the wealth of this kind of online expression out of hand; I will instead merely point out that the genres of Web-based authoring are very conservative, after more than a decade of the Web, and the forces leading toward the one-button publishing model (which basically limits user-level Web authoring to forms like blogging) are strongly in ascendance.

Is this just nostalgia for a golden age? Before standardized platforms, so much more was possible—this is of course a truism. No doubt there is an element of nostalgia in my analysis, but let us remain focused on the black boxes, and which ones we might benefit from keeping open.

Why Not Smalltalk? Why Not the Dynabook?

We are today the heirs of two dominant traditions of computing. First, the Unix tradition is manifest in our Internet technologies and in much of the programming/development culture—especially that of the FOSS movement. The second is a "personal computing" tradition which can be traced academically back to Xerox PARC, but which bears more the stamps of the industrial and marketing forces that shaped the personal computer as a commodity in the 1980s and 1990s. These two traditions are in significant tension with one another; the Unix tradition prizes depth of knowledge, inner workings, and almost delights

in its arcana. In this tradition, software is more important than hardware. Its practitioners, especially in the FOSS movement, value a do-it-yourself ethic and the free sharing of the fruits of their efforts, which become the scaffolding for others' work. The personal computing tradition, on the other hand, has become a consumer-oriented tradition; here, (buying) hardware is more important than software. It is in many ways a discourse of anxiety: how to deal with complexity, inundation, threats, spam, viruses? The answer is to buy the next version, the latest offering. These two traditions are at this point entirely intertwined: the Unix tradition relies on the ubiquity of cheap hardware and network access provided by a personal computing market; the latter relies on the former for the network itself, the means of using it, and, in a broader sense, innovation. The web browser is the canonical interface between these two worlds.

There is, as I have attempted to demonstrate, a third tradition—or at least the seeds of one—nascent, latent, possible. This is the tradition of the Dynabook. It differs markedly from the two dominant traditions today, and as such it offers antidotes to the major failings of each. What the Dynabook offers to the Unix tradition—with which it is contemporaneous and shares many key virtues—is a *superior vision of the user*. Alan Kay's key insight in the late 1960s was that computing would become the practice of millions of people, and that they would engage with computing to perform myriad tasks; the role of software would be to provide a flexible medium with which people could approach those myriad tasks. Unix, in constrast, has always been a system for systems people;[21] despite the democratic rhetoric of the FOSS movement, Unix has no serious potential to become the computing environment of the masses; it is not difficult to see the cultural/historical dynamics working directly against this.

Relatedly, what the Dynabook offers to the "personal computing" tradition is also a superior vision of the user, but in this instance, the difference is that the Dynabook's user is an *engaged participant* rather than a passive, spectatorial consumer—the Dynabook's user was supposed to be the creator of her own tools, a smarter, more capable user than the

21. To be fair, Smalltalk-80 has also been primarily a system for systems people.

market discourse of the personal computing industry seems capable of inscribing—or at least has so far, ever since the construction of the "end-user" as documented by Bardini & Horvath (1995).

This is all simple enough. Why has the Dynabook vision not prevailed? Or, prevailing aside, why is it so very marginalized? The short answer is elaborated in the "sociology of knowledge" literature. It is fatalistic in a historically contingent way: the circumstances surrounding the emergence of these computing traditions, with their associated virtues and vices, led to a particular historical unfolding; once cemented in their extensive networks (marketing, manufacturing, popular discourse, journalistic coverage, and the pedagogical process of initiating new project participants), the 'ecological niche' possibilities for other options diminished. This process is nicely described in the "social shaping of technology" literature (e.g., Bijker & Law 1992), in which the manifold possibilities of a new technology are completed and subsequently reified by social factors. This also works well with the notion of "irreversibility" of techno-economic networks raised by Michel Callon (1991). Personal computing has thus become "a network whose interfaces have all been standardized," and therefore which "transforms its actors into docile agents and its intermediaries into stimuli which automatically evoke certain kinds of responses" (p. 151). Is that enough? Perhaps not.

To end the story—and the analysis—here is to miss much of the rich and interesting detail; it is to revert to the agnostic stance, in which technocultural phenomena are judged 'behaviourally' by their empirical historical impact or by the extent of their propagation. It is to miss—or willfully ignore—the questions of what makes a *good idea* good, a *powerful idea* powerful. What makes us able to recognize a good or powerful idea, and conversely what prevents us from that recognition? Why should we want to sidestep those issues? We have to go deeper, into the specifics of the thing.

The *technocentric* answer to the question of why the Dynabook vision has not prevailed is one of implementation engineering: the Unix (and by extension the programming language C) tradition makes a virtue of prying computing efficiencies—and therefore

speed—out of language abstraction. C strikes an extremely different bargain with a programmer than Smalltalk does. C (and like languages: Pascal, C++, etc.) allows a trained programmer to achieve computing *efficiencies* not too far off writing in assembly language (i.e. very low level, close to the machine) while giving that programmer a reasonably expressive language. In contrast, Smalltalk (and Lisp before it) are languages which *begin* with abstraction; they are mathematically derived, and the implementation details follow; more important is the conceptual flexibility and scaffolding opportunities afforded the programmer. The result is that though they allow vastly greater and more flexible abstraction and open-endedness, dynamic languages like Smalltalk and Lisp simply produce slower-running software than languages like C, all other things being equal. This issue, and the debate over whether it is preferable to conserve processor time or programmer time has existed since the early days of Lisp. But, more practically, where this issue has hit the marketplace, especially in the early days of microcomputers, the need for low-level performance has dominated.[22]

A more interesting and culturally situated treatment of this cultural divide is explored from within the Lisp community in Richard Gabriel's (1991) article, "The Rise of 'Worse is Better,'" which suggests that the virtues of *simplicity*, *correctness*, *consistency*, and *completeness* are in different proportion in different systems design communities, and, by extension, that the definitition of these qualities changes somewhat according to their interrelationship. Gabriel identifies the Lisp school of design (the "MIT approach") as *the right thing* with a particular formalist rendering of these four qualities, putting *correctness* and *consistency* foremost. The "worse-is-better" school of design, however, places a higher prority on the virtue of *simplicity*, which brings about a refiguration of the other three:

> Early Unix and C are examples of the use of this school of design, and I will call the use of this design strategy the *New Jersey approach.* I have intentionally

22. Recall that while Alan Kay was at Apple Computer, it was all they could do to wring barely usable performance levels out of 1980s-era Macintoshes running Smalltalk. Their experience hardly inspired anyone to adopt such technology, and they continued to use it—even to the point of developing a specially optimized version (Smalltalk-V) for the Vivarium Project—despite the practical issues of performance.

> caricatured the worse-is-better philosophy to convince you that it is obviously a bad philosophy and that the New Jersey approach is a bad approach.
>
> However, I believe that worse-is-better, even in its strawman form, has better survival characteristics than the-right-thing, and that the New Jersey approach when used for software is a better approach than the MIT approach. (Gabriel 1991)

Without getting into the details of Gabriel's analysis, and to say nothing of the generations of rebuttals and counter-rebuttals he and others have written, it is informative to see that the issue can be quite effectively addressed from *the inside* of a particular cultural context, and that such expositions tell us something quite different from an external analysis. That morally charged vocabulary like "right thing" and "correct" can be used unproblematically (or at least constructively) in a consideration of why technological systems tend toward one or other historical trajectory is key: this is a long way from the "social shaping" approach, in that it allows for the virtues of the ideas themselves—or at the very least their incarnation in systems—to speak alongside more straightforward social and cultural factors. It speaks to significance and meaning-making that is emergent from technocultural systems and their attendant virtues.

That said, there is nothing here to suggest that there is only one "insider" perspective; perspectives are emergent; they rise and fall, as discursive patterns come and go, are reinforced and reiterated or wither and fade away. Gabriel's discussion is not an argument for C and Unix, but rather an assessment of the fate of Lisp in a C/Unix-dominated world. This assessment has a particular character today which is decidely different from what it looked like in 1975 or 1985. This is an *ecological* shift, though, as opposed to an argument being won or lost on objective merits. Note that even a technocentric analysis, taken to sufficient depth, necessarily becomes one of ecology.

The Dynabook: existence and essence

The Dynabook—like Smalltalk itself—is no more or less marginal an idea today than it ever was, despite there being better supports—technological and cultural—for it than there ever

have been before: computer hardware is inexpensive and ubiquitous; the Internet is widespread and more or less faithful to its original concept; there exists now a set of norms and institutions governing the widespread sharing of software; and, though this way of expressing it has fallen out of favour, "computer literacy" is a matter of practical concern to millions of people, young and old alike. The extension of *networks*—in both the literal sense and in Latour's sociotechnical sense—is far greater and more powerful. If there was ever a time for the Dynabook to succeed in the world, surely it is now.

But the cultural milieu—about ideas and meanings and the relative significance of things—are far from favourable to the Dynabook today. What is recognizable as a powerful idea has shifted significantly. What counts as a virtue or a vice in this cultural milieu has shifted since the mid 1970s. On some level that is not at all technical—nor even technocultural—the Dynabook seems very far apart from the values and concerns and practicalities of today's world, today's schools, today's students. I can argue here that this is an unfortunate thing, but there is little I can do to bridge the gap conceptually.[23]

In his evocative and stylish case study, *Aramis: For the Love of Technology*, Latour goes so far to as to raise the question of whether or not Aramis, the ill-fated rapid transit system, *exists*:

> Chase away the people and I return to an inert state. Bring the people back and I am aroused again, but my life belongs to the engineers who are pushing me, pulling me, repairing me, deciding about me, cursing me, steering me. No, Aramis is not yet among the powers that be. The prototype circulates in bits and pieces between the hands of humans; humans do not circulate between my sides. I am a great human anthill, a huge body in the process of composition and decomposition, depending. If men stop being interested in me, I don't even talk anymore. The thing lies dismembered, in countless pieces dispersed among laboratories and workshops. Aramis, I, we, hesitate to exist. The thing hasn't become irreversible. The thing doesn't impose itself on anyone. The thing hasn't broken its ties to its creators. (Latour 1996, p. 123)

23. DiSessa articulates a similar theme with reference to the difficulty of obtaining funding for computational media projects in a world in which so many of these black boxes are closed. "The self-evident state of the art blinds people to other possibilities" (2000, p. 241).

Surely the ontological status of the Dynabook is no different. Aramis too comprised many powerful ideas, but powerful ideas alone didn't make it real. Despite three decades of almost constant attention by Kay and his circle of devotees and developers, the Dynabook exists only tenuously: implied in the marginal survival of Smalltalk, echoed in our wireless laptops, remembered by a small cadre of developers, but surely in danger of total collapse as soon as guiding hands fall away.

Or is it? Does Latour's parable of the irrevocable, tragic *reversibility* (*pace* Callon) of technosocial systems totally apply? At some point, the Dynabook's continued play for existence (if not actual existence) over thirty-five years puts it in a different class than the much shorter-lived Aramis project. Does this make the Dynabook more alive—or just more *undead*, a better candidate for the B-movie horror genre? And what of the myriad facets of the Dynabook which have been realized and even reified in any number of successful forms, from the various Smalltalks to Powerbooks, Photoshop, and Peer-to-Peer networks? Bits of the Dynabook surely live on—though mostly far removed from educational concerns. But if these surviving elements are no longer connected to an educational project, let alone an educational vision, then we have to ask whether the Dynabook has been translated into unrecognizable form, into non-existence.

The bigger question which emerges here is perhaps not in what ways does the Dynabook experience resemble or not resemble Latour's account of Aramis. The bigger question, it seems to me, is whether Latour's technosocial ontology has everything it needs to account for a phenomena like the Dynabook. Latour, for one, seems to have no time for anything resembling a "powerful idea"—such things, when they appear at all in a story like Aramis' seem to come and go, are either engineers' fantasies or mere epiphenomena of Latour's concrete, materialist philosophy. Such materialism makes for tidy technosociologies. I am not convinced, however, whether it does justice to the experience of technosocial actualities.

Where this story needs to go next is on to the Dynabook's latter-day resurgence, in a project called Squeak, which arose in Kay's final days at Apple Computer in the 1990s, and

which has carried his project along ever since. Whether the Dynabook is real—in any meaningful sense—surely has something to do with the reality of Squeak.

Chapter 7: Squeak's Small but Mighty Roar

> *In the back of our minds, we all felt that we were finally doing what we had failed to do in 1980.*
>
> – Dan Ingalls, 1997

The story related so far is unremarkable in so far as it tells of the rise and fall of a cultural object—an idea whose time has come and gone; a technology which has become obsolete; an ideological or philosophical stance no longer relevant in a changed world. If the story were to end there, this finality would be an easy conclusion, and we would be in a position, like Latour in *Aramis*, to ask—in the past tense—who or what killed the Dynabook?

What makes this story more complicated and ultimately more important to present-day concerns of educational technology, the political landscape of the digital sphere, and even approaches to the sociology of knowledge is the significant re-emergence of the Dynabook project from Alan Kay's team in the late 1990s: a new Smalltalk environment called *Squeak*. Not merely a re-release of "classic" Smalltalk, Squeak represented a re-contextualization of Smalltalk in the age of the Web, multimedia, and the free/open-source software movement. Squeak symbolizes at least the persistence of the Dynabook vision and exists as a kind of proof of concept for the applicability of Kay's 1970s-era research in a later age. The Squeak project and the communities surrounding it provide interesting fodder for an actor-network approach to technocultural history; on the one hand, Squeak's trajectory and the challenges of establishing networks of support and currency make for straightforward actor-network exposition; on the other hand, the extent to which Squeak embodies ideas which have been lurking, latent, without supporting networks, for close to two decades, presents challenges for this mode of sociology; it is difficult to account for Squeak and its relative success from a purely materialist frame.

Squeak: A Renaissance Smalltalk

By 1995, Alan Kay's momentum had slowed considerably. The Vivarium Project, which had provided a development and research focus through the 1980s, had "run down" by 1993, and the software research that followed it amounted to little. The patronage of Apple CEO John Sculley ended in 1993. After a decade of struggling to regain the energy of the 1970s, the mid 1990s must have been a frustrating time. I saw Kay deliver a keynote speech at the third World-Wide Web conference at Darmstadt in 1995; wielding his videotape of Douglas Engelbart's 1968 demo in which the software pioneer showed a mouse-driven, networked hypermedia system, he chided delegates not to be so proud of the Web, and made an appeal for a network of message-passing objects instead. After his talk, he was set upon by dozens of young developers dying to know what he thought about Java, Sun Microsystems' brand new OOP language, positioned very much as a Web technology. Kay remained guarded.

The same year, Dan Ingalls, who had been the chief architect of Smalltalk at Xerox PARC, returned to Kay's team. Ingalls had come to Apple Computer in the mid 1980s to do research work on Smalltalk at Apple, but this lasted only a short time. Ingalls left Apple, and the IT industry entirely for almost a decade. In the early 1990s, he took a job at Interval Research working on a Smalltalk-based home-media project. In late 1995, Ted Kaehler began coaxing Ingalls to come back to the team at Apple. Ingalls' work at Interval required him to come up with a Smalltalk implementation, which he did using the relatively open-licensed Apple Smalltalk and the program listings published in Adele Goldberg and Dave Robson's 1983 book *Smalltalk-80: The Language and its Implementation*—the book which had been the culmination of Xerox PARC's release of Smalltalk into the wider world. Musing over a reunion with Kay and Kaehler at Apple, Ingalls felt that he would need to re-create a Smalltalk version again, and, in a flash of insight, he realized that what he had done at Interval could be accomplished *mechanically*. Ingalls joined Kay's team at Apple in late 1995, and immediately began this work.

"Back to the Future"

In a 1997 conference paper by Ingalls and a few of his long-time associates, he wrote:

> In December of 1995, the authors found themselves wanting a development environment in which to build educational software that could be used—and even programmed—by non-technical people, and by children. We wanted our software to be effective in mass-access media such as PDAs and the Internet, where download times and power considerations make compactness essential, and where hardware is diverse, and operating systems may change or be completely absent. Therefore our ideal system would be a small, portable kernel of simple and uniform design that could be adapted rapidly to new delivery vehicles. We considered using Java but, despite its promise, Java was not yet mature: its libraries were in a state of flux, few commercial implementations were available, and those that were available lacked the hooks required to create the kind of dynamic change that we envisioned.
>
> While Smalltalk met the technical desiderata, none of the available implementations gave us the kind of control we wanted over graphics, sound, and the Smalltalk engine itself, nor the freedom to port and distribute the resulting work, including its host environment, freely over the Internet. Moreover, we felt that we were not alone, that many others in the research community shared our desire for an open, portable, malleable, and yet practical object-oriented programming environment. It became clear that the best way to get what we all wanted was to build a new Smalltalk with these goals and to share it with this wider community. ("Back to the Future." Ingalls et al. 1997)

This "new Smalltalk" was *Squeak*, released (from Apple Computer's Research Labs) in October 1996 as Dan Ingalls made a short announcement on the comp.lang.smalltalk Usenet newsgroup: "Squeak—A Usable Smalltalk written in itself" (Ingalls 1996). The "back to the future" phrase was particularly apt; not only had Alan Kay managed to regain his core Xerox-era technical team in Ingalls and Kaehler, but they were able to pick up where they'd left off, nearly two decades before.[1] The Squeak project is notably *not* a necrophilic dredging of the glory days; its prime motivation was a profound dissatisfaction with the tools availa-

1. Ingalls later reflected on the split (between an educational focus and systems programming focus) that seems to have emerged at Xerox PARC after Smalltalk-76 was created. According to Ingalls, Kay had suggested at the time that the more child-friendly Smalltalk-72 could be implemented *within* Smalltalk-76, and Ingalls agreed. "But the thing is, I didn't do it." With Squeak, first priority was given to educational and personal media tools like animation, sound, and child-friendly programming.

ble, as well as the knowledge that what they had had in the late 1970s had not been equalled. Kay had spent, by this point, over a decade trying to wrestle Apple's systems into service for his educational projects, using various 1980s-vintage Smalltalk implementations, simulation environments like Agar and Playground, and Apple's own tools (like HyperCard), without much success. In 1995, Sun Microsystems' Java appeared, drawing in large part on Smalltalk's vision: a network-aware, object-oriented system that could serve as an interactive multimedia platform for the Web. But Java failed to capture Kay's imagination; it was far too inelegant for him: "Java is the most distressing thing to hit computing since MS-DOS," he reportedly said at a 1997 conference (Guzdial 1997).

What Kay's team at Apple wanted was clear enough: something as elegant as the Smalltalks they had worked with at Xerox, but brought up to date, with the modern Internet and modern multimedia in mind (colour, sound, video). They needed something like Java, but not the huge, byzantine system that Java was emerging as. Java was distressing because it delivered on some of the promise—a network-aware object-oriented system—but without the tidy conceptual elegance (e.g., the kernel expressible in "half a page of code") which could make it usable (and extensible) by people other than professional software engineers. Of course, Java's popularity soared, and with it its complexity. Ingalls' aesthetic addressed this, too; he wanted to write a Smalltalk entirely in Smalltalk, to make the system completely self-contained. This would accomplish two very important things: it would allow the system to be capable of evolution: if the base implementation needed to change, this could be accomplished 'from within.' Ingalls wrote Squeak *in* Apple Smalltalk-80, and as a result, *future versions can be written within Squeak*. But second, having a self-contained Smalltalk implementation would mean that it was vastly more "portable": Squeak could be implemented on a Mac, on a PC, on a Unix workstation, or what have you.[2] Ingalls' goal was to create a *bootstrappable* Smalltalk that would free it from dependency on any other platform—or vendor.

2. Ingalls' paper, "Back to the Future" details exactly how this "Smalltalk written in itself" was accomplished. The practical results are impressive: the team at Apple had Squeak working in a mere 16 weeks, (Ingalls et al. 1997). When the code was released onto the Internet, ports to Unix and Windows appeared within 3 and 5 weeks, respectively (Ingalls 2001).

The larger context for this development was the fate of Apple Computer, which in 1996 could not have looked darker. After long-time CEO John Sculley's departure from the company in 1993, Apple's corporate health was in crisis. By 1996, frequent senior management changes, big layoffs, and mismanagement had pushed Apple's stock price lower than it had been in 10 years, descending into a trough that it would not recover from until 2000. While it has been fashionable for computer industry pundits to predict Apple's imminent demise ever since the company's founding, at no time did it look more likely than the mid 1990s. Microsoft had just released its enormously popular *Windows95* operating system and was in a period of unprecedented growth. Many people—including Apple insiders—believed that Apple's days were numbered.

The Precariousness of Smalltalk in 1996

Whether Apple Computer would collapse or not, Kay's research funding was in jeopardy, and the climate at Apple was grim (Rose 2001, part 1). Furthermore, the rest of the industry was at this time looking at Apple like a risky partner. Squeak developer John Maloney recalls working with Kay's team at the time:

> *Constructo and several earlier projects had been built using Digitalk Smalltalk/V, but Digitalk had just stopped supporting Macintosh. The Power PC [cpu chips] had just come out and it was clear that there was not going to be a PowerPC-based Digitalk Smalltalk. Since we were at Apple, there was no question that whatever we built had to run on Macintoshes. So Digitalk was a dead end. There were also commercial Smalltalk systems from IBM and ParcPlace. However, IBM's Smalltalk did not run on the Mac and ParcPlace required hefty licensing fees. As I recall, the official ParcPlace licensing fee was about $2000 per user—way more than schools could afford. Adele Goldberg always claimed that if we wanted to test stuff in schools she would make a way for that to happen, but without a clear path for wide-scale, royalty-free distribution of our software, we didn't want to put a lot of effort into ParcPlace Smalltalk. (Maloney 2001)*

By throwing in their lot with Apple, Alan Kay and his colleagues had in a sense chosen sides in the 'computer wars' of the 1980s; it had begun as an Apple vs. IBM battle, but by the mid 1990s, Apple's adversary was clearly Microsoft. By 1996, this must have looked like a dangerous predicament. The development of Squeak—a *portable* implementation of Smalltalk—and Kay's work to secure an *open source* license agreement for it ensured that they had an escape pod. Though the development work had been done at Apple, and had relied largely on Apple resources, Squeak would be in no way dependent upon Apple; if Apple were to 'go under' and Kay's team to find a new home, at least the fruits of their efforts would escape with them.

As it happened, Kay and his research team did indeed jump ship; In October 1996, just as Squeak was publicly released on the Internet, Kay and his colleagues left Apple to join Walt Disney Imagineering, recruited by the head of R&D there, film and theatre designer Bran Ferren. Kay and the Squeak team would remain at Disney for another 5 years. Apple Computer, of course, would survive to make blue computers and iPods for a waiting world.

Squeak's release to the Internet was, in a sense, the culmination of a project begun in 1979, when Adele Goldberg and colleagues set about preparing *Smalltalk-80* for release to the world beyond Xerox PARC. But the computing industry of 1980 had evolved limited options for the general release of technologies. Goldberg's aspiration for Smalltalk was for it to be released as widely as possible, and in fact, a number of the original licenses were sold for $1 (Goldberg 1998). However, the entanglements between Xerox Corporation, Parc-Place Systems (the company spun off to license Smalltalk-80), and the limited number of interested licensees) meant that Smalltalk-80 remained a marginal technology. In 1996, however, with a widely available Internet and a growing movement in free and open-source software, Squeak had the opportunity to have a truly wide reach.

The initial results were encouraging. By the end of 1996, external developers had sight-unseen picked up Squeak and ported it to Windows and Unix platforms, confirming Ingalls' belief in its platform independence (the October 1996 release had been a Mac application). Today, Squeak runs on more varied computing platforms than almost any open source technology (over 20, according to viewpointsresearch.org).

Portability and openness were just two of the virtues Squeak embodied; also key was the fact that Internet and multimedia capabilities were almost immediately available within Squeak. In 1997, a system was released that allowed Squeak to run within a web page (by way of a web browser plug-in, just like Java, Flash, and other multimedia platforms), so Squeak-based content could be accessed via the Web. This makes Squeak much easier to access for teachers and others restricted from installing software on lab computers.

From the standpoint of the core developers, Squeak meant that they were once again in control of their software. Georgia Tech's Mark Guzdial notes that, "Alan Kay is most proud

that each generation of Smalltalk obliterated the version before, until it went commercial. Then it became frozen. Use Squeak as a way to obsolete it"(Guzdial 1997). Kay hoped that Squeak, though based on Smalltalk-80, would open up the development of the architectures and concepts underlying the language itself. In his introduction to Guzdial's book, *Squeak: Object Oriented Design with Multimedia Applications*, Kay wrote:

> In Squeak, you have in your hands one of the most late bound, yet practical, programming systems ever created. It is also an artifact which is wide, broad, and deep enough to permit real scientific study, creation of new theories, new mathematics, and new engineering constructions. In fact, Squeak is primed to be the engine of its own replacement. Since every mechanism that Squeak uses in its own construction is in plain view and is changeable by any programmer, it can be understood and played with. "Extreme play" could very easily result in the creation of a system better than Squeak, very different from Squeak, or both.
>
> We not only give permission for you to do this, we urge you to try! Why? Because our field is still a long way from a reasonable state, and we cannot allow bad defacto standards (mostly controlled by vendors) to hold back progress. (Kay 2000*b*, p. xii)

This very idea that Squeak might be the instrument of its own obsolescence speaks to an important tension within the Squeak community.

Squeak as an Educational Platform

> *Squeak Is An Idea Processor For Children Of All Ages!*
>
> – Alan Kay, *www.squeakland.org*

Within the Squeak community that has emerged since 1996 is a tension which dates back to the 1970s at Xerox PARC: a system of sufficient flexibility, concision, and conceptual elegance to be useful as an educational media platform is also a powerful draw for systems developers and professional programmers. This tension was manifest in the translation of Smalltalk-72 into the more robust Smalltalk-76, and it is manifest in Squeak, which presented itself first as a personal media toolkit along the lines of Kay's Dynabook vision (in

Kay's terminology, an "almost new thing"), but which was quickly picked up by members of the Smalltalk development community as an open-source Smalltalk implementation ("a better old thing"). Today, these two communities are clearly distinct, with different websites and mailing lists serving as the touchstones and meeting places of these groups. For instance, the website at squeak.org and the *squeak-dev* mailing list primarily serve the development community; a different website at squeakland.org (and the *squeakland* mailing list) address the concerns of educators and learners (there are also a large number of other distinct subcommunities online—more about that below). This unresolved tension has existed since the 1970s at Xerox: certainly since Kay's "burn the disk packs" turning point with Smalltalk-76, and probably long before that (Ted Kaehler, personal communication); what is different today, however, is that both the systems focus and the educational focus draw on three decades of prior work. In practical terms, this means that Squeak as an educational platform is certainly not starting from scratch.

Etoys: Doing with Images makes Symbols

Etoys emerged in the early years of Squeak, and is practically synonymous with Squeak for many people who encounter it. Etoys is a tile-based scripting environment for children that lets them orchestrate sophisticated behaviour for onscreen objects—without having to compose programming code in the traditional way (i.e., typing syntactically perfect code and then watching what breaks). Etoys, thus, is a latter-day iteration of Kay's evolving ideas about kids as novice programmers, building on a three-decade heritage of research in this area.

This work began in Smalltalk-72 at Xerox, with the Learning Research Group investigating what kinds of scaffolding are required in order for kids and novices to achieve meaningful results. Adele Goldberg's "Joe Box" work is the hallmark of this research, generalized to the idea that a set of well thought-out superclasses as building blocks then allows users to specialize them in interesting ways. The original Smalltalk-72, drawing on the Logo language, attempted to offer a very simple syntax and keep the referents concrete, but the

mode of user interaction was still primarily textual—one typed in program code one line at a time. Smalltalk-76 furthered the idea of a system architecture based on a hierarchy of object classes and provided a host of graphical tools for accessing those components, but this was arguably of more use to systems developers than children or novices.

Kay wrote a chapter for *World Book Encyclopedia's Science Yearbook 1979* entitled "Programming Your Own Computer." This chapter laid out a schematic for a programming interface re-oriented to novice users (indeed, high-school students were the audience for this book). In the *World Book* chapter, a "player" object could be inspected via a very simple window displaying the current values of various parameters, and these values could be directly manipulated or scripted. The importance of such a feature is that it brings the "direct manipulation" interface (Laurel & Mountford 1990) to the realm of programming; by clicking on an onscreen object, one can reveal properties and behaviours, which can then be directly modified.

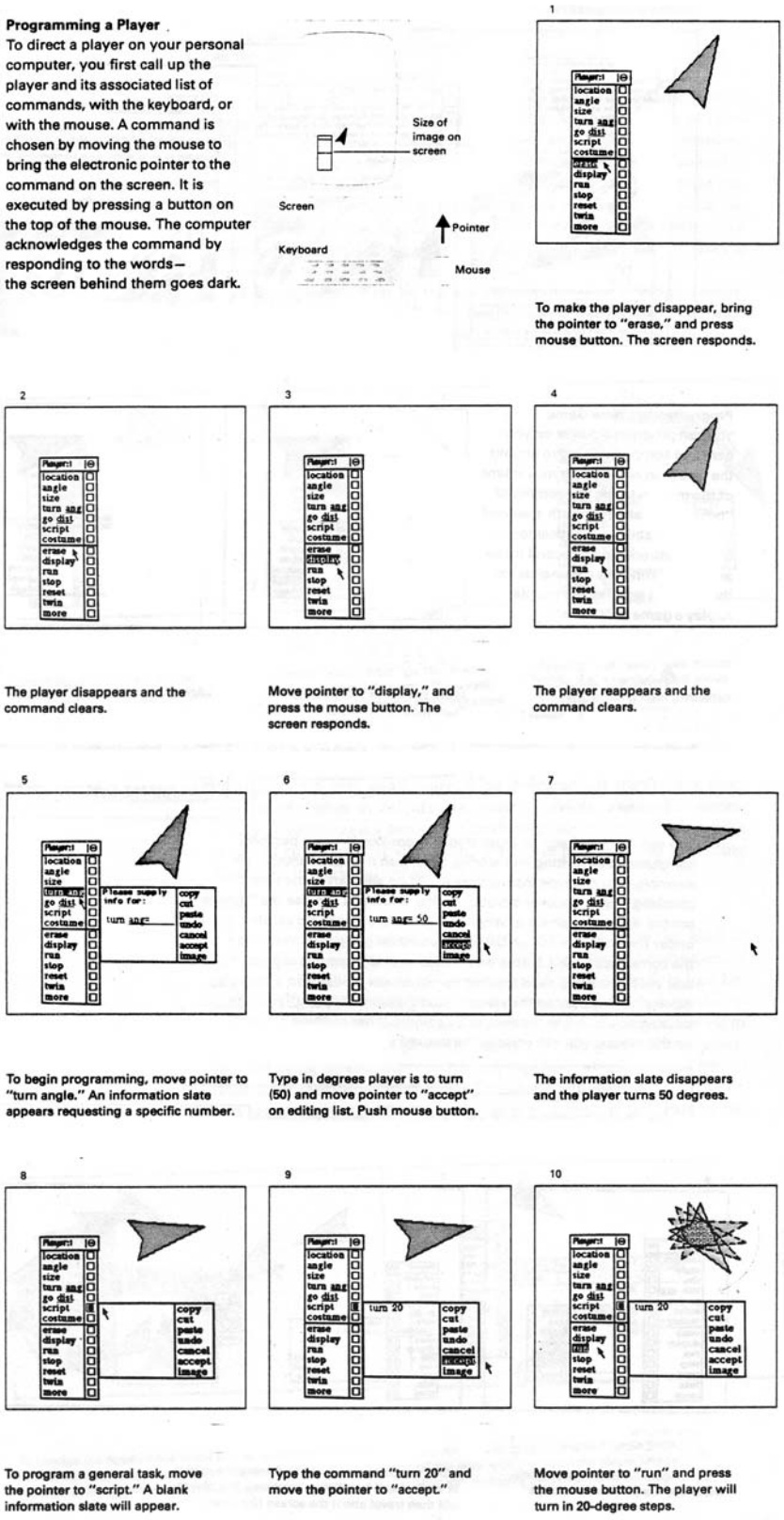


Figure 7.1: From "Programming Your Own Computer" (Kay 1979)

The programming interface in the *World Book* chapter was idealized, but Kay's projects in later years would realize this ideal. The *Playground* system from the Vivarium project at Apple was based on "player" objects that could be inspected in just this way. "One way to think about Playground is as having a spreadsheet view, a HyperCard view, and a textual programming view, all simultaneously," wrote Ann Marion (1993, ch 1. p. 1).

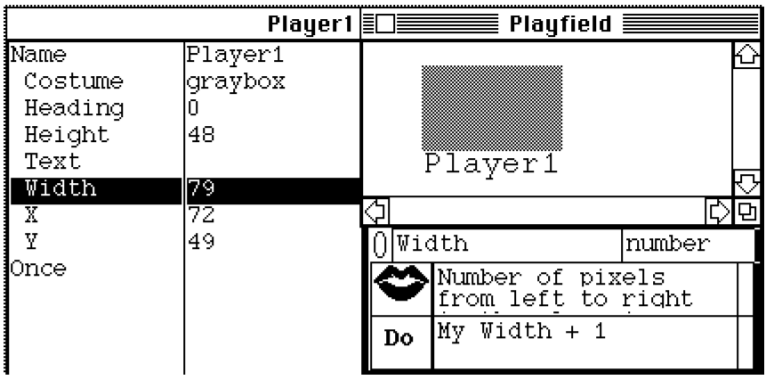


Figure 7.2: Playground II "Scriptor" (from Marion 1993, ch. 1 p. 7)

But Playground, with its "observer" model inspired by the dynamic spreadsheet (as opposed to the Smalltalk message-passing model) proved a bit too rigid. Playground "players" were not message-sending objects, but observers, watching other objects' parameters in the same way a formula cell in a spreadsheet *watches* the values of the cells it depends on. For example, in the Playground environment, a simulated 'shark' object is continually watching the parameter values on a 'prey' object; a message-sending version of the same thing could have the 'shark' object specifically querying the 'prey' object for its co-ordinates or direction, or, alternatively, the 'prey' object sending out various parameter values, which may or may not be received by the 'shark' object. The "observer" model produces an arguably simpler, cleaner programming environment, but provides a single, limited model for inter-object communication. Kay recollected:

> Playground was kind of a generalized event-driven system that had objects which were kind of like a collection of generalized spreadsheet cells,

> completely concurrent, etc. This gave the kids the state of objects to look at, but not call/return. Every value was a thread.
>
> I loved to program in it, but I thought that, in the end, it was a little too pristine for 9 year olds—it was a little too much like pure Lisp functional programming in how clever you needed to be. The cleverness was rewarded by beautiful, simple, and powerful programs, however. (Kay 2001)

Etoys in Squeak built upon this player-and-viewer model, merging parts of the "observer" idea with Smalltalk's message-passing model. It also drew substantially from a tradition of *iconic programming* research at Apple and elsewhere. Iconic programming—that is, composing program-like structures by manipulating graphical elements on screen—dates back to Ivan Sutherland's *Sketchpad* and the RAND Corporation's early pen-based *GRAIL* system, and has been sought after ever since. Kay mentions having iconic programming in mind in the first Dynabook concepts (Kay 1996*a*, p. 552). David Smith's *PYGMALION* system ("an executable electronic blackboard"), which dates from the mid 1970s at Xerox, is an early and celebrated iconic programming environment (Smith 1993). Smith and Allen Cypher later worked on an iconic programming system called *Cocoa* while at Apple Computer (Cocoa was commercialized in 1999 as *Stagecase Creator*). And, during his brief stay at Apple in the mid 1980s, Dan Ingalls (along with Scott Wallace and others who would later create Squeak) created a Smalltalk-based iconic programming environment called *Fabrik*. An Apple "Advanced Technology Group Research Note" (Chesley et al. 1994) entitled *End User Programming: Discussion of Fifteen Ideals* illustrates the kind of thinking that was active in the mid 1990s: throughout, direct manipulation and visual concreteness are key virtues.

Kay's touchstone was his reading of Jerome Bruner's model of three distinctive modes of thinking: *concrete*, *iconic*, and *symbolic* (Bruner 1966, p. 11ff; Kay 1990, p. 195ff)—that these are not developmental stages we pass through so much as styles of thinking which we use in different combinations in different contexts. The way forward, according to this thinking, was not simply to replace the symbolic with the iconic—that is, to replace program code with iconic representations—but to recognize and nurture the interplay

among all three. A key reminder of this was the straightforward success of Bill Atkinson's HyperCard, which, while being much simpler (and more restrictive) conceptually than any of the Smalltalks, facilitated simple scripting by attaching messages to buttons or text fields on the screen. A user could use graphics tools to create an onscreen object, thereby making something concrete to work and think with—and *then* attach some scripting or message-sending to it. This simple insight was taken up by Kay's team for Etoys (Ted Kaehler, personal communication, October 2004): allowing direct manipulation of objects (that is, onscreen with the mouse) to lead to scaffolded scripts, one line at a time, and then to further generalized programming at the point when it seems easier to move to the more abstract form.

What Etoys provides is a visual programming environment that is a blend of the direct-manipulation iconic style and the textual/symbolic approach. Any graphical object (Etoys encourages one to begin with a graphic object) can be scripted by opening a "Viewer" very much like the one sketched out in the 1979 *World Book* chapter; this presents a number of the object's attributes (position, heading, etc.) to be shown in a list of direct-manipulation windows.

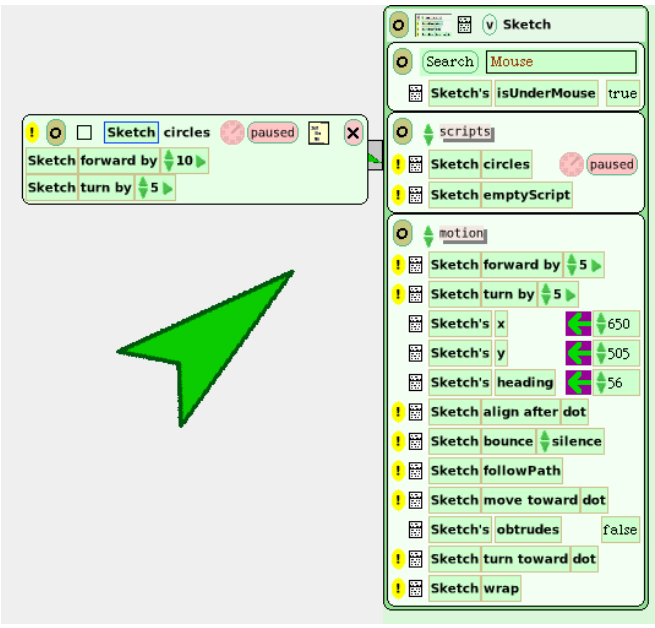


Figure 7.3: Etoys "Viewer" in Squeak 3.8

At the most immediate, the values of the various parameters can be changed, either by typing or by choosing possible values from a menu. Some of the parameters are animation oriented: for each "tick" of a system clock, any parameters concerned with changes—position, heading, etc.—will update by the specified amount. The result is that simply by clicking and typing in the Viewer window, the object can be made to change and move immediately. The second step is to drag a "script" window out from the Viewer. Once this has been done, "tile-based" programming is available. Parameters from the Viewer can be drag-and-dropped into the script; here is where the tradition of iconic programming intersects with Etoys. With a script constructed, running the "ticker" system clock causes the script to be executed (looping is the usual behaviour). The script can be modified as it is running by dragging new items from the Viewer, or removing them, or by changing the values of object parameters. Conditional "tests" can be dragged in as well, again, using the Viewer parameters as components. So a good deal of programming functionality can be accomplished simply by dragging and dropping parameters and script components, and by directly manipulating the parameters in real time (this also provides a real-time update of the various parameters). With just what I have described, most of Logo-style turtle geometry can be accomplished via direct manipulation: no "code" needs to be written as such.

The *Morphic* User Interface Framework

The technology that makes dragging and dropping of script components work in the Etoys environment is due to Squeak's primary GUI toolkit, called *Morphic*. Morphic was originally developed as part of a blue-sky research environment called *Self* at Sun Microsystems in the late 1980s and early 1990s. Self was designed to be a completely direct-manipulation programming environment; its goal was to leapfrog Smalltalk in terms of conceptual simplicity ("Self is like Smalltalk, only more so," wrote its original designer, David Ungar).

John Maloney, who had worked on the Self project while at Sun, joined Alan Kay's team at Apple in 1995, and brought the Morphic interface with him; he re-implemented it in Squeak, and in 1997 Morphic became the foundation of Squeak's—and Etoys'—user interface.

Morphic has been a double-edged sword for Squeak. While it is arguably one of the most malleable graphics environments ever developed, and while it supports the ideal in Squeak that absolutely everything in the system should be manipulable, this may also be a drawback for newcomers to the system. Indeed, much of the criticism of Squeak from the computer science community is centered around the bewildering collection of layers and components which confront the user looking at Squeak (see WikiWikiWeb: MorphicInterface).

However, since Etoys is merely an interface to Squeak Smalltalk, there is a traditional Smalltalk code representation lurking in each Etoys object as well; and it is possible to toggle between the draggable "script" version and a more traditional code editor window.

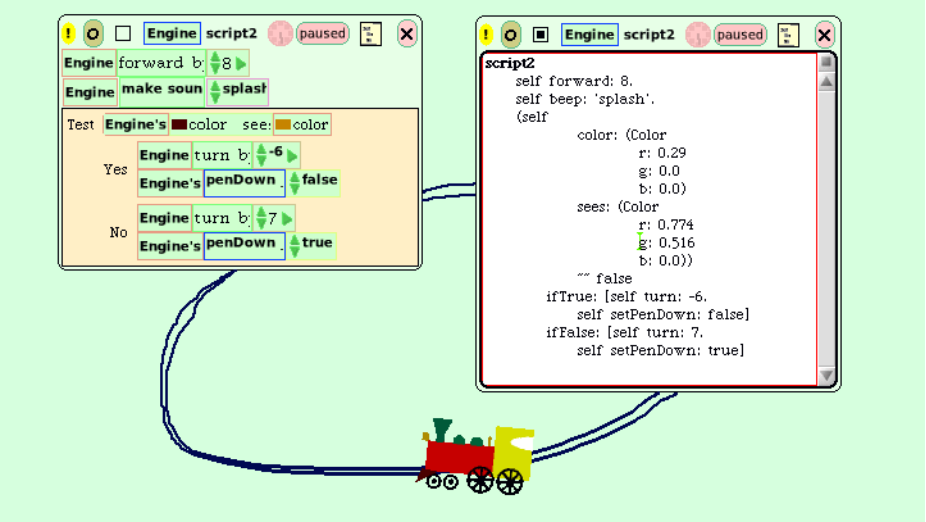


Figure 7.4: Etoys tile representation and equivalent Smalltalk code

The effect is that Kay's Bruner-esque "Doing with Images makes Symbols" is embodied most directly in the Etoys interface: concrete, direct manipulation is the most immediate kind of interaction made possible here; iconic tiles then can be used as stand-ins for black-boxed functionality (such as the Logo-derived "move forward 5 steps", or even, at a higher level of organization, an entire script represented as an icon or tile), and the symbolic level is available as well. The idea is that kids can encounter and explore the Etoy environment in any (or any combination) of these styles.

Squeak vs. Squeakland

Is Etoys what Squeak is all about? Yes and no; it is undoubtedly the central focus of Squeakland.org, the website and community established by Alan Kay's nonprofit Viewpoints Research Institute in Glendale, CA. Squeakland is the public face of Squeak in educational contexts, and Etoys is what it is all about. As we shall see, there are other faces to Squeak, but this one forms the core of the present discussion.

The Squeakland.org website offers about two dozen Etoy *projects*—these are downloadable Squeak files which can be accessed either via the Squeak web browser plugin or from a full Squeak installation. A "project" is an encapsulated multimedia artifact, typically consist-

ing of an Etoys simulation or tutorial (often as an "active essay"—that is, explanatory text accompanied by a number of active Etoy components). These downloadable projects are organized into rough curricular divisions: elementary school, middle school, high school. Some have been created by Viewpoints staff, but others are showcased works by children or teachers.

B. J. Allen-Conn, a teacher at the LA Open School, and Viewpoints executive director Kim Rose have published a book of Etoys exercises (Allen-Conn & Rose 2003) which perhaps serves as the central text of the Squeak educational community. The book walks though—tutorial style—a series of Etoys exercises which deal in a roundabout fashion with the concept of acceleration. The book begins with the usual Etoy starting point:[3] painting a small graphic of a car. The car object is then made to move around the screen by adjusting the parameters in the Viewer. Next, we are instructed to paint a steering wheel, and parameters for the steering wheel are connected to those of the car, so that when we rotate the steering wheel object, the car turns left and right. This basic Etoy can be built in a couple of minutes, and is presented as an introductory Etoys exercise.

> The "Montessori game" here is for the children to get most of their pay-off playing in the hand-eye arena, while gradually and subliminally gaining fluency and appreciation of the power of symbols. (Allen-Conn & Rose 2003, p. 7)

3. The Etoys "car" exercise is featured as well in several of Alan Kay's recent lectures and talks; it serves as the basic Etoys demo.

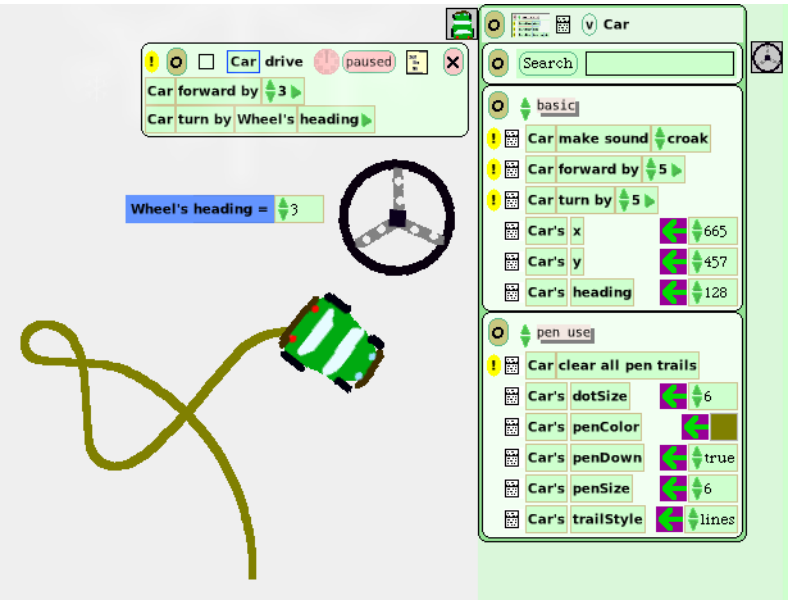


Figure 7.5: The hallmark "drive a car" Etoy, with dynamic interaction between two objects, represented by the two square tabs in the upper right; the Viewer panel for the Car tab is open and visible here. Tiles such as "Wheel's heading" can be simply dragged and dropped as building blocks into a script (Allen-Conn & Rose 2003).

From this direct-manipulation exercise, we are invited to begin scripting the car to behave semi-autonomously; following an arbitrary track, and speeding up and slowing down. Then, two cars can be raced on the screen, and the concept of acceleration is introduced by distinguishing between speed (distance travelled per 'tick') and change in speed (one parameter affecting another parameter). With a model for visualizing acceleration established, Allen-Conn & Rose then turn to a discussion of gravity; we are invited to go outside and drop different kinds of weights from a height, as with Galileo's classic experiment. Squeak is re-introduced as a multimedia tool at this point, to examine frames of digital video shot of the dropping weights. By measuring the distance travelled by the falling weights from frame to frame, a concrete visualization (an "inscription," in Latour's terminology) of acceleration is re-created; now the earlier simulation model of acceleration can be adapted to a simulation of gravity's pull on falling objects (pp. 69–71).

In Allen-Conn & Rose's treatment, Etoys seems aimed at teachers and schools; mostly upper elementary and middle school. Kim Rose (personal communication, 2004) reports

that she works with a small (30-odd) but growing number of schools on a regular basis, helping to support their use of Etoys; this represents a good proportion of the traffic on the *squeakland* e-mail list.

But interestingly, when Etoys were first developed, schools were *not* the target; rather, individual children (and parents) surfing the Internet were. Alan Kay explained it this way:

> I should say a little about the history of Etoys. They were originally not aimed at classrooms but as 10–20 minute projects supplied on the web for parents and their children to do together. I stripped out as many features as I could and tried to come up with a system that could do "100 examples" pretty straightforwardly. The documentation that was intended here was to have been to teach parents how to do the examples so they and their kids could have a good experience. For several reasons, this plan did not work out at Disney. But BJ [Allen-Conn] saw it and wanted to try Etoys in her 5th grade classroom. I was initially against the idea because I thought that Etoys were not complete enough for that venue. But she and Kim Rose decided to do it anyway. Six weeks later they started to show me some really good results, and I realized that it would be worth doing a 3 year experiment to see how well the Etoys—even with some of their lacks—would work out with 10 and 11 year olds. (Kay 2003*b*)

Squeak in School

Squeak was not originally aimed at *schools*; indeed, homeschooling is a theme lurking just under the surface of the Squeak philosophy.[4] Introducing Squeak to educators was not originally a priority. Even the Etoys user interface, which represents a significant advance in terms of kids having some leverage in a full-featured programming environment, was thought of in its early years as something that curious children would find on the Internet and become interested in. However, once BJ Allen-Conn picked up Squeak for use at the LA Open School, Squeak became a school-based technology. And, one must admit, if one's goal is to reach children, schools are at least a good place to find them (Kim Rose, personal communication, Oct 2004).

4. The Squeakland website lists "unschooling" advocate John Holt as one of its "Deep Influences on Thinking and Learning" (along with Bruner, Piaget, Montessori, and others).

Squeak's open-source license is somewhat telling. By making Squeak an open, non-commercial entity, Kay and team abandoned much of the traditional world of vendor-driven educational technology. Squeak was aimed at children first, and then perhaps teachers. In contrast, 'successful' educational technology projects seem to be aimed at adminstrators and technology co-ordinators first. As such, Squeak has been spreading via word-of-mouth rather than via a marketing campaign as such (Kim Rose, personal communication, Oct 2004). It is, however, extremely difficult to say anything definitive about the extent of Squeak's use by individual learners, parents, or even teachers, in the same way it is difficult to say how many workstations running Linux there are in the world. What we can talk about with some certainty is what particular teachers have been doing with Squeak.

After the Vivarium Project ended at Apple Computer (around 1993), BJ Allen-Conn maintained close contact with Alan Kay and Kim Rose, and continued to be part of their circle for many years. Every summer for many years, Alan Kay has hosted a "Learning Lab" retreat at the Apple Hill Centre for Chamber Music in rural New Hampshire (the attendee lists read like a who's who in educational technology[5]). At the 1997 retreat, Allen-Conn saw an early version of Squeak. At the time, her technology curriculum at the LA Open School relied on an amazingly widely varied suite of tools: *Logo*; Apple's *Cocoa* (later *Stagecast Creator*, a graphical "programming by demonstration" environment); *AgentSheets* and *HyperGami* (simulation software and 3D-modelling environment from the University of Colorado); and Amy Bruckman's *MOOSE Crossing* (a constructionist MOO environment from the MIT Media Lab)—all these *simultaneously*.[6] But after seeing Squeak at Apple Hill, she told Kay she wanted it for her classroom. Despite Kay's initial reservations about using Squeak in a classroom setting, Kay and Rose and colleagues set about getting a version ready. Allen-Conn began to replace the various software tools she was using, one by one, with Squeak. The factor that made the difference for her, she says, was the amount of time

5. For more info about Kay's Apple Hill retreat, see http://minnow.cc.gatech.edu/learninglab/17
6. BJ Allen-Conn's ongoing connection to cutting-edge educational technology research is probably unparalleled among classroom teachers.

the children spend problem-solving, compared with wrestling with the software (Allen-Conn, personal communication, Nov 2004).

Since that time, Allen-Conn has represented the vanguard of educators using Squeak as a curriculum tool, and one of the primary developers of Squeak/Etoy-based curriculum resources (such as the book she and Kim Rose published in 2003). Similiarly, the LA Open School has remained Kay's educational testbed, although not at the scale it enjoyed while Apple was funding whole-school, integrated curriculum research in the 1980s. Other notable users of Squeak in school settings include two schools in Toronto, where school district trustee Sheine Mankovsky is a vocal advocate (Don Mills Middle School and Don Mills Collegiate).

Beyond North America, it would appear, Squeak has more of a foothold. As the old saying goes, Squeak is "big in Japan." In fact, at a 2003 conference in Kyoto, "over 300 teachers, students, researchers and developers gathered ... to spend a day dedicated to talks and sharing about Squeak, curriculum development, best practices and new Squeak media." These numbers must be compared to the 2004 and 2005 Squeakfest conferences, held in Chicago, and which attracted 80 and 60 people respectively.[7] There is also Squeak activity in Korea and Brazil. The largest numbers may be in Europe; a substantial community of developers is found in Germany. In Spain, the regional government of Extremadura has undertaken a very large-scale project to install open-source software in schools. Over 60,000 computers are running Squeak on Linux there, though information about actual use of Squeak in classrooms is harder to come by.[8]

Despite Squeak's geographical distribution, though, the number of schools and teachers actually using Squeak is, frankly, small. There seems to be more 'interest' in Squeak as an educational environment than there is actual practice. The *squeakland* mailing list, which is specifically for educational topics, has very little traffic: 400+ subscribers, but only a handful of postings per week. By comparison, the *squeak-dev* list, which serves the programmer

7. Information about the Japanese Squeak conference from http://squeakland.jp/images/news/html/kyotonews.htm—the US-based Squeakfest can be found at http://interactive.colum.edu/partners/squeakfest/part.aspx
8. http://www.small-land.org

community, has 1358 subscriptions as of the fall of 2005, and commonly runs to 20–30 messages per *day*.

Why isn't Squeak a bigger deal, given its heritage, its flexibility, its goals, its cost? The reasons are not surprising, but they are worth outlining in some detail:

- *Technology support in schools*: The LA Open School is far from typical in its commitment to technology, for reasons both ideological and financial. BJ Allen-Conn is quite skeptical of the idea that Squeak could spread significantly via 'viral marketing' in a self-sustaining way. "Teachers need support," she insists, in the form of books and guides and one-on-one support; somebody has to teach the teachers, and make it possible—if not easy—for a teacher to actually pick up and use something like Squeak in their classroom. In order for Squeak use to be more widespread, the supporting resources—books, tutorials, installers, etc.—would need to be much more robust. Allen-Conn sees herself as the 'buck-stops-here' element of Kay's Squeak team: she has to make it meaningful for children: "I can think lofty thoughts, but I also have to make this a viable tool." (personal communication, Nov 2004). So, despite the fact that Allen-Conn herself remains encouraged, Squeak's presence simply will not grow of its own accord.
- The *difficulty of sustaining technology once installed*: Allen-Conn points out that during the 1980s, the entire teaching staff at the Open School was trained to maintain and repair Apple computers. In the years after Apple's involvement, the parent-teacher community of the Open Charter School has taken it upon itself to keep up its committment to the sustainability of its technical investments. How many schools can claim this kind of involvement?
- *Technology policies in most schools and school districts are notoriously restrictive*: for support and administrative reasons, the typical pattern is that classroom or lab computers are 'locked down' to prevent students from making modifications that may put a burden on the tech support staff. So, installing software other than that

officially approved (typically Internet Explorer, Microsoft Office, and a small handful of 'educational' applications), neither students nor teachers are at liberty to install anything else—like Squeak. Going beyond these blanket restrictions would likely entail getting approval at an adminstrative or even board level, something that even companies with substantial marketing budgets struggle to do. Again, this is an instance in which the general autonomy of the Open Charter School provides a very atypical example.

- *General awareness and mainstream trends*: more generally, Squeak faces an uphill battle simply by virtue of its distance from the mainstream (i.e. commercial software and/or Unix-derived Internet culture). If educational computing is commonly defined as learning to use a particular canonical list of application programs (web browsers for research, 'office' software for document production, domain-specific games or simulations for particular pedagogical ends), what are teachers or coordinators to make of an environment that at times presents itself as programming language, operating system, animation tool, simulation environment, or that bills itself grandly as an "idea processor whose music is ideas?"
- *Squeak is probably more complex than it needs to be:* allowing a somewhat technocentric criticism here, it is fair to point out that the Squeak environment that one faces after first launching the software is not immediately obvious: this undoubtedly presents a practical challenge to Squeak's wider adoption. First, Squeak doesn't look or act quite like a Macintosh or Windows application, drawing as it does on both older (and newer) user interface traditions; users must grapple with how to interact with the mouse and onscreen objects before moving on to more interesting things. Second, there is substantial *clutter* in the Squeak environment; much of the thirty-five years' heritage of Squeak is actually *in there* (hundreds of object classes, multiple interface frameworks, components of software no one has actively used in years, and so on). The *shrinking* of the downloadable Squeak

"image" is the subject of ongoing debate on the *squeak-dev* mailing list.[9] And finally, the *Morphic* user interface framework itself, with its complete transparency and direct-manipulation access to *everything*, has the potential to overwhelm and confuse new users.

Despite Squeak's three-decade heritage, Etoys' features, and the educational virtues espoused by Kay and his colleagues, Squeak still remains an unknown quantity to mainstream educators, even those who are active in the world of educational technology. Is this likely to change? Not given the current landscape. There are simply no substantial forces driving Squeak toward the educational mainstream, and there are many forces actively resisting it. But we must keep in mind that the educational face of Squeak, of which Etoys is emblematic, is not the whole story.

The Squeak Community and its Trajectories

What to make of Squeak? To whatever extent it represents an important step forward, or the next iteration of a powerful tradition, it is also in many ways a non-starter, a tiny blip on the vast landscape of (educational) technology. It seems as though one must grapple with the history and heritage of the Dynabook ideal in order to fully appreciate Squeak; that without this historical perspective it appears as a bewildering and complex software environment, inordinately difficult to sum up, let alone to treat as an instrument of practical consequence. The hope of many in the Squeak community, presumably, is that Squeak should gradually build its own momentum, and thereafter we will have a better appreciation of its intellectual heritage.

But what is Squeak, exactly? Any singular answer to this question is partial, simply by virtue of the multi-headedness of the project. I have so far focused mostly on the educational (Etoys) face of Squeak, but this is not manifest destiny. On the contrary; on the

9. The default downloadable Squeak image, as of version 3.8, is about 13 megabytes. There are various opinions about how small an image could be while still being usable by the average person. One of the more extreme projects, called *Spoon*, has an image of less than 100 kilobytes. See http://www.netjam.org/spoon/

Internet, an open-source project becomes what its users and developers make of it; there is no ontological directive, apart from the actual practice of the people who choose to make a committment of time and energy to a project. In this sense, the lessons from Latour's Aramis are most apt: "There is no such thing as the essence of a project. Only finished projects have an essence" (Latour 1996, p. 48). But when is a project ever finished? In Squeak's case, the sheer flexibility of the system means that there are various sub-communities making headway in a number of different directions at once. Squeakland and Etoys represent only one of these.

The Blue Plane and the Pink Plane

> *There are two orthogonal forces at work in the Squeak team, with which we have been able to make two kinds of progress. These have most recently been articulated in Alan Kay's allusion to Arthur Koestler's metaphor of progress in two planes: the incremental improvement plane (which Alan calls the "pink" plane) and the paradigm shift (or "blue") plane.*
>
> – Dan Ingalls, 1997

The metaphor of planes of progress is drawn from Arthur Koestler's *The Act of Creation* (1964), one of Kay's touchstones: Koestler's model bears some resemblance to Thomas Kuhn's dynamics of paradigm shifts, though Koestler was concerned with creativity as a psychological process rather than with the workings of institutional science. In any case, these two 'forces' manifest themselves as two distinct but not wholly separate cultures active in the Squeak community. The distinction can be characterized as follows. Incremental improvement, or "pink plane" development, is the focus of a group of developers who treat Squeak as a "better old thing," as Kay might put it—that is, Squeak is a new, portable, open-source Smalltalk-80 implementation. The efforts of this group go towards making it a better implementation; improving the speed, cleaning up the codebase, putting in infrastructure support for the development community (bug-report databases, versioning systems and collaborative tools, etc.), streamlining and enhancing the user interface, and adding various applications. This group, composed largely of Smalltalk programmers who have been working with the language for years, behaves much like the development

communities surrounding other open-source projects, like Linux or Apache. There is no central leader of this subcommunity; rather, there are a small number of key developers who communicate largely via the *squeak-dev* mailing list, co-ordinating the changes which periodically constitute new version releases of the Squeak software, with incremental updates appearing roughly twice per year.[10]

The more revolutionary, "blue plane" development appears to happen in a less open environment. This force is represented more by Kay's development team and colleagues. Prior to the v3.4 release of Squeak in 2003, this "Squeak Central" team—that is, the group at Apple and later Disney—made a number of substantial changes to Squeak (mostly notably integrating the *Morphic* graphics framework, which was called a "blue plane" innovation at the time), but since this time they have not been particularly active in the online Squeak development community. Rather, they have been working "behind the curtain" on a number of emerging innovations; notably, a new novice programming environment called *Tweak* (which may also include the future of Etoys), and an immersive, multiparticipant 3D environment called *Croquet*. These projects may potentially shift the focus of Squeak development substantially, even to the point of threatening the relevance of some of the "pink plane" development going on among the open-source community. Dan Ingalls notes:

> To best understand the "blue" pulls within the Squeak group, you need to understand what we're after. Our number one commitment is to an exquisite personal computing environment. Imagine a system as immediate and tactile as a sketch pad, in which you can effortlessly mingle writing, drawing, painting, and all of the structured leverage of computer science. Moreover imagine that every aspect of that system is described in itself and equally amenable to examination and composition. Perhaps this system also extends out over the Internet, including and leveraging off the work of others. You get the idea—it's the Holy Grail of computer science. All and everything. So if some new approach comes along that takes us closer to that ideal, but at the cost of a

10. There is a complex history of leadership within the Squeak community, which I will not go into in detail. In 2005, a group called The Squeak Foundation, composed of several influential developers from around the world, declared itself the (un)official steward of Squeak releases. See http://www.squeak.org

break with ST-80 tradition, we will probably take the new approach. (Ingalls 1997)

To put it in Eric Raymond's terms, the Squeak project is simultaneously the "cathedral" and the "bazaar." Kay, Ingalls, and their cohort work in semi-private, releasing sizable new innovations only now and then, while the evolutionary group works according to the bazaar model. There is clearly some tension between the two groups, particularly when the "cathedral" releases appear, and the evolutionary group must grapple with the changes in the landscape.[11]

Squeak Communities Today

After Kay's team left Disney in 2001, there ceased to be a central commercial interest behind Squeak. Kay and Rose established the nonprofit Viewpoints Research Institute to act as the official home for Squeak, but while Viewpoints has been partially funded in recent years by Hewlett Packard,[12] the mainstream of Squeak development is no longer centered there. It is rather spread globally, in a wide variety of different projects. The result is, as evidenced by the potpourri of links on the main *squeak.org* website, that there are several public faces of Squeak currently:

1. The *Viewpoints Research Institute*, the non-profit organization which is officially behind Squeak, and which acts as an institutional focus for some of its development. Viewpoints is the interface between the funding that Squeak may receive from a company like HP and the notion of an open-source project owned and beholden to no one. In practice, Viewpoints acts as the co-ordinating force behind *Squeakland.org* and "blue-plane" development emerging from Kay and his immediate colleagues (e.g. the new *Tweak* authoring system, which purports to replace portions of the Morphic interface).

11. This issue comes up periodically on the *squeak-dev* mailing list. In September 2005, for instance, a discussion of the introduction of a technology called "traits" as an alternative to the nearly 30-year old hierarchy of object classes spawned much debate.

12. In July 2005, following a year in which Alan Kay received no less than three high-profile awards for his contributions to computer science and engineering, Kay and VPRI colleagues (along with 14,500 other employees) were 'let go' from HP in a round of corporate restructuring.

2. *Squeakland.org*, hosted by Viewpoints Research Institute: this is the home (or at least the central meeting place) of Etoys and the web-browser plugin version of the Squeak software. On this website can be found a number of articles by Kay and others elaborating the philosophy of Squeak, Etoys projects and tutorials created by the core team, and links to Etoys projects and tutorials contributed by teachers and students. Squeakland.org can be thought of as the "teacher-friendly" face of Squeak. The accompanying *squeakland* mailing list is low-volume (a handful of messages per week), and the topics of discussion are generally user-oriented.
3. The *squeak-dev* mailing list, which is the operating vehicle of Squeak's open-source community. Squeak-dev is a high-traffic list (hundreds of messages per week) with over 1350 subscribers. The topics of discussion on this list are very technical: this is the forum for discussions of Squeak's core programming, upgrades, bug reports and fixes. The members of this list think of Squeak as first and foremost a Smalltalk implementation. This community also finds expression in a number of web-based forums and blogs such as *planetsqueak.org*, *squeakpeople.org*, and Mark Guzdial's *SWiki* at *minnow.cc.gatech.edu*.
4. The *Croquet Project* (*croquetproject.org*) has the potential to radically change the public face of Squeak. Croquet is an immersive, multiparticipant 3D environment built on Squeak. The Croquet Project is maintained by Viewpoints Research Institute, the University of Wisconsin, and the University of Minnesota. Croquet has an educational profile, but it seems much more relevant to the post-secondary world than to K–12 (see Lombardi 2004). Croquet seems content to brand itself distinctly from Squeak, and although Squeak is the underlying technology, a casual browser could peruse the rich and informative croquetproject.org website and miss the reference entirely.
5. Alternative educational applications to Etoys, such as Stephane Ducasse's *Bots Inc.* (see Ducasse 2005 and *http://smallwiki.unibe.ch/botsinc/*), and the *Scratch* (see

Maloney et al. 2004 and *http://weblogs.media.mit.edu/llk/scratch/*) project at the MIT Media Lab—part of the lab's *LifeLong Kindergarten* program, and aimed at the Media Lab's after-school Computer Clubhouses. Both *Bots Inc.* and *Scratch* are combinations of Squeak-based software and curriculum resources.

6. *Seaside*, a highly sophisticated Web application framework built in Squeak.[13] Seaside breaks significantly from much of the Squeak tradition in that it uses Squeak in service of the Unix client-server model; the user interface in this case is provided by your web browser. Seaside may be an atypical Squeak application, but as Web development matures (e.g. the "Web 2.0" movement currently afoot), Seaside's profile is quickly growing *outside the Squeak community*.
7. Beyond these key English-language sites and communities, Squeak is actively used and developed in Europe, Asia, and South America, both as an educational application (with Etoys) and as an open-source Smalltalk. The *Squeak Foundation*, for instance, is (nominally, at least) based in Germany, as is *Impara*, a company co-founded by core Squeak developer Andreas Raab. There is a also strong Japanese school-based effort at *squeakland.jp*. I do not wish to give the impression that Squeak is entirely an American phenomenon.

What ties these varying *brands* together? Not much, currently. Despite the fact that they all draw from a common ancestor, and all embody in some way some combination of the virtues of the Dynabook vision, there is no clear sense that these different themes are converging or even diverging. A more unified brand would perhaps create a stronger image to present to the world beyond the initiates, but the proliferation of versions presumably allows Squeak to be picked up in different forms in different community contexts. Originally, Etoys was aimed at children surfing the Internet, or at least their parents—that is, it

13. Coincidentally, Seaside's lead developer is Avi Bryant, who developed Ricki Goldman's *Orion* back in 2000 (not in Seaside, though). In 2006, Bryant's company, Smallthought, released a web-based data management application called *DabbleDB* that leverages the strengths of Smalltalk to seriously push the envelope of web-based applications. For a discussion, see http://www.techcrunch.com/2006/03/11/dabbledb-online-app-building-for-everyone/

had a deliberately subversive trajectory. But, largely due to BJ Allen-Conn's success (and the book she and Kim Rose published in 2003), Etoys has become the mainstream educational face of Squeak; other efforts now take on the subversive, *viral marketing* agenda. The immersive, 3D Croquet environment, with its obvious appeal to video-game players (and lack of an overt 'educational' message), has the potential to take this on, even given its backing by major US universities as a platform.

Squeak in print?

Ironically, the success of Squeak in the wider world depends greatly on the publication of paper-and-ink books. What has made Squeak legitimate in educational contexts is largely the availability in the early 2000s of a number of books on Squeak; the first significant one was written by Georgia Tech professor Mark Guzdial (2000); the second was a collection of essays edited by Guzdial and Kim Rose (2003); the third and perhaps most significant is Rose and BJ Allen-Conn's (2003) *Powerful Ideas in the Classroom*, a book targeted specifically at teachers—"helping the helpers," as the Viewpoints motto goes. Similar books have been published in Japan, in France, and in Spain. Stephane Ducasse's *Squeak: Learn Programming with Robots* (2005) follows a number of French-language books.

Having books in print adds strength to the project in a number of ways: first, the mere existence of a book lends credibility; the open-source movement has known this for years, and with any new project, the emergence of books from the key technical publishers (especially O'Reilly or one of the Pearson imprints: Prentice-Hall, Addison-Wesley, Peachpit, Que, New Riders[14]) is a sign of some maturity. Second, though, a book represents a vector for newcomers to be able to pick up the software and make sense of it without having to pick through the maze of often incomplete online documentation that accompanies most open source projects (for that matter, most software, open or not).

The need for printed resources presents a challenge for Squeak, which has no central body pouring resources into it. Neither HP (which until recently employed Kay, Rose, and a

14. Note the "obligatory passage point" here in establishing legitimacy even among open-source projects.

number of key developers) nor the Viewpoints Research Institute (which is a nonprofit society, and devoted much more to development than to marketing) is really in a position to underwrite the writing and publication of books (or to realize any sort of return on them). In 2005, The Squeak Foundation was established; one of its motivations is to provide an entity through which development funds could flow. Notably, author Stephane Ducasse was a driving force behind the establishment of the Foundation.

So far, though, the Squeak community produces books slowly, and though everyone seems to recognize books as hallmarks of maturity, it would appear that no one is in much of a position to strategically address this. And perhaps it is also evident that this is a good thing. Were an Apple Computer or Disney Imagineering to wholly back a project such as Squeak, what would be the cost in terms of control of the project's direction? There is a very good case to be made for keeping Squeak independent of any one corporation's interests, despite the attendant lack of resources.

The variety of foci within the Squeak community is a factor here as well. On one hand, this can be interpreted as a sign of maturity, and could give rise to a stronger sense of confidence in the ongoingness of the development. On the other hand, however, it could be seen as the fragmentation of an already small and marginal project. But suppose a popular book on Croquet were to appear. If Croquet really took off, would Etoys wither? Perhaps not, given its tenacity in the face of its small user base to date, but this does raise the question of the required "critical mass" for a project to begin to generate its own momentum.

Where is that Dynabook, Anyway?

Given the growth of the Squeak project—perhaps it is even a stretch to characterize it as a single project anymore—are we any closer to the Dynabook vision? A recurring theme discussed on the *squeak-dev* mailing list is the idea of porting Squeak to the latest, smallest handheld or palmtop device on the market.[15] The reasoning here is that Squeak, running on a device which looks something like Kay's cardboard mockups (circa 1970), must equal the

15. The technical windfall here is that Squeak has been ported to over 20 different hardware and software platforms.

Dynabook. And while, following the letter of the text, putting checkmarks next to all the required elements in Kay's 1972 manifesto, this might look like the real thing, it is, like all literalist accounts, depressingly ahistorical, as it ignores what personal computing has come to mean in the intervening 35 years. The sobering counterfact is that *no one* I have encountered is able to actually conduct all of their daily work in Squeak; I visited Viewpoints Research Institute in 2004 and found the same general clutter of technologies and platforms there as everywhere else—this despite the existence of web, e-mail, and text processing applications in Squeak. So despite Squeak's ambition to be an "exquisite" personal computing environment, in practical use it is an application like the rest.

Does Squeak bring us closer to the Dynabook vision, then? Absolutely, if for no other reason than it has re-animated the virtues inherent in the original 1970s vision and spread them to a much larger and more varied community of users and developers spread all across the world. Squeak itself has not commanded the attention of a mass audience—and certainly not within educational circles—but perhaps the various bits and pieces of Squeak have an aggregate—possibly subversive—effect.

The question which Squeak begs—is it really "back to the future?" Or is it "too little too late?"—cannot quite/yet be answered in its own terms. Only finished projects have an essence, Latour tells us. It would appear, on the whole, that the amount of energy within the Squeak communit(ies) is increasing, not decreasing. Whether or not this ultimately matters is a question for future historians.

Squeak and Croquet at OOPSLA'04

In 2004, Alan Kay's star seemed to be on the rise. Early in the year, he (along with PARC colleagues Chuck Thacker, Butler Lampson, and Robert Taylor) was awarded the *Charles Stark Draper Prize* from the US National Academy of Engineering. The Draper award is given to "those who have contributed to the advancement of engineering and to improve public understanding of the importance of engineering and technology" (NAE website)—in this case, the award was in honour of their work on the Xerox Alto, "the first networked

personal computer." Later that spring, two other awards were announced, beginning with the *2004 Kyoto Prize* by the Japanese Inamori Foundation, in honour of "those who have contributed significantly to the scientific, cultural, and spiritual betterment of mankind" (Inamori website)—this specifically with respect to Kay's educational work. This year's Kyoto laureates (alongside Kay) were biologist Alfred Knudson and philosopher Jurgen Habermas.

The third award was the Association of Computing Machinery's *Turing Award*, given to those who have made "contributions of a technical nature made to the computing community ... of lasting and major technical importance to the computer field." The Turing Award is a prize of unparalleled prestige within computing circles; it is usually attended by a "Turing Lecture" given by the recipient at an ACM function. Kay chose to deliver his lecture at the ACM's Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) conference in Vancouver, in October 2004. I took the opportunity to attend, and brought my notebook.

Interestingly, the OOPSLA conference was established in the early 1980s by Adele Goldberg, who, after leaving Xerox PARC, served for a time as the president of the ACM. OOPSLA was, in the 1980s, the conference for the Smalltalk community. In later years, Smalltalk became a smaller and smaller part of OOPSLA, with the rise of object-oriented systems based on the languages C++, Java, and more recently, Microsoft's C# and .NET architecture. The 2004 conference program was dominated by technical papers on professional software engineering concerns, though with a tiny undercurrent of, if not Smalltalk itself, at least some of the practices that the Smalltalk community had pioneered, such as *design patterns* and *eXtreme programming*.[16]

Alan Kay gave two talks at OOPSLA'04: his *Turing Lecture* (Kay 2004*c*) plus a keynote address to the "educators' symposium" track of the conference. Both talks featured demonstrations of both Etoys and the 3D Croquet environment. The Etoys demonstrations served

16. The *design patterns* movement is an application of architect Christopher Alexander's "pattern language" (Alexander et al. 1977) concepts to software design. *eXtreme Programming* is a software development methodology which originated in the early Smalltalk community of the 1980s; see Beck 2000.

to point out to the audience (professional programmers largely paid to manage and construct systems of enormous complexity) how very simple implementations can acheive great power and flexibility. Croquet served to demonstrate the range of what could be done with a relatively simple system like Squeak. It is difficult to tell what kind of impact Kay's talks had on their audiences. Perhaps Kay's impassioned address on behalf of simplicity and elegance in the face of (arguably needless) complexity strikes a chord among the listeners. I was aware, sitting in the audience with members of the Squeak *illuminati*, of a sort of conspiratorial glee over Kay's critique of the mainstream, relished by the Squeak and Smalltalk "in crowd," but, just as likely lost upon or dismissed by the much larger audience of professional software engineers.

The following day, I attended an open-mic session billed as "breakout ideas in computer science." Speaker after speaker lined up to address the hundred-odd person crowd on what he or she (there were relatively few women in attendance, but the proportion was stronger in this particular session than in others I attended) felt was wrong or missing in their contemporary software development environments and careers. A surprising number of complaints (the session quickly became an airing of grievances against the field) spoke of the need for innovations which had been in Smalltalk, and in Kay's and Ingalls' and others' visions, since the 1970s but of which even this 'mainstream' remained ignorant.

The speakers—computer programmers—complained of inhuman systems for managing complexity, of having to bend their own minds and practices around rigid and inflexible systems. One speaker went so far as to comment that what the world of programming needed was the equivalent of the Macintosh, which had allowed end users to do great things. He wondered "what would the equivalent be for general-purpose programming?" I hope the reader at this point recognizes the irony of these comments, especially at an event honouring Kay's contributions to these very concerns.

During Kay's *Turing Lecture*, he presented a rough schema in which the general population is broken down along two axes, as follows:

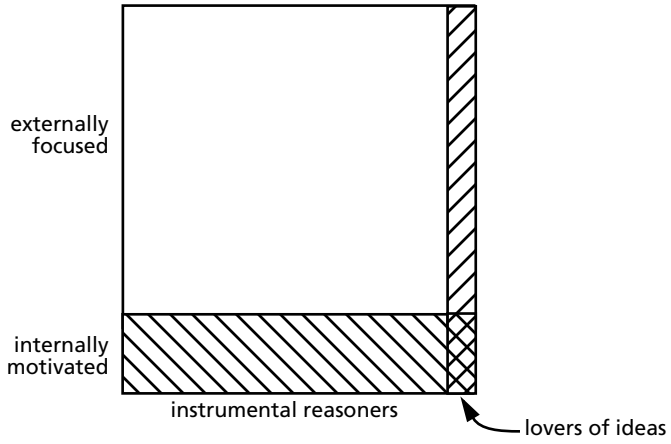


Figure 7.6: Kay's schema from the 2004 Turing Lecture.

According to Kay's model, 95% of the population are driven by *instrumental reason*—that is, the application of tools and technologies to pre-existing goals—while the remaining 5% are motivated by a *love of ideas*. On the vertical axis, Kay suggested, 15% of people tend to be *internally motivated* while 85% are *externally focused*. Mapping these two against each other, Kay pointed out the 1% overlap: internally motivated, idea-driven people who occupy the "creative, investigative space." Next, there are 4% of people who are externally focused and idea-driven; these are popular leaders and charismatic figures. On the other hand, the 14% who are internally motivated instrumental thinkers are "a dangerous lot" whom Kay identified with certain corporate executives. Finally, Kay claimed, are the 80% of the population who are externally focused instrumental reasoners, and these are the people who make up the vast majority of the population. These people don't change their thinking easily—unless they begin to perceive that everyone around them is changing, too.

It is, at face value, a decidedly pessimistic schema, and one that suggests that real reform—in education, or computer science, or politics, or anywhere—is not something that can be acheived, in many cases, within any project or even one person's lifetime. It is, however, a schema that Kay seems to have made peace with; he explicitly acknowledged that the kinds of reform he wanted to see would not likely happen while he is around to

witness them. This does not apparently dampen his enthusiasm or energy for the task, though.

Squeak: Mouse that Roared?

I alluded earlier to the question that Squeak's now eight-year existence begs: is it really, as Ingalls et al. suggested, *back to the future*, or—given the sheer inertia of the dominant tradition of personal and corporate computing, and the relatively tiny size of Squeak's communities (both developers and educators)—simply *too little too late?*

Squeak is many things. It is, undeniably, the resurrection of the Smalltalk project from Xerox PARC, especially given the personnel behind it. Squeak is also, undeniably, a very cool and fun open-source software project. I am tempted to go so far as to call Squeak the most powerful piece of educational software ever created, given its scope and its direct connection to some of the most "powerful ideas" in educational technology.

But Squeak is not the Dynabook. It is no more the Dynabook than the Smalltalks of the 1970s were. It is better conceived as the culmination of or the wrapping up of these projects. I would suggest that the "Squeak Central" team's departure from the core development of Squeak in 2003—in favour of more far-reaching projects like Croquet—is evidence of this wrapping-up process. There is nothing particularly ironic or surprising about this; Kay has maintained all along that Squeak's goal was to be the means of its own obsolescence, the vehicle for *getting to the next thing*. In searching for the Dynabook, therefore, we are advised not to spend too much time looking at Squeak itself.

That Squeak—on its own in the world, the public property of a distributed, open-source development community—has not emerged as the "next great thing" tells us something important about the dynamics of technocultural systems. To take Squeak as important in itself is to attempt to *isolate* it, to *reify* it, to *prevent its translation* into something else. This is something very different from the case of Latour's *Aramis*, which struggled *to be*, to cross some kind of threshold into actual, sustainable existence. To contexualize Latour's admonishment that no technosocial assemblage is truly irreversible—not even the "100 year old

monsters" of the Paris Metro—the objects of his analysis were intended to be stable, irreversible instruments; their ultimate and ongoing success would be defined by their unfailing resistance against being translated, yet again, out of existence.

By contrast, Squeak and the Smalltalks which preceded it in a sense succeed by their very mutability. It is not an accident that Kay chose to name the original system "*small talk*." This is not a technology, nor a technocultural assemblage, that aims to resist being translated into something else. Rather, this is an entity whose ultimate *telos* is to be the *agent* of that translation. It's the very point.

It is instructive to recall Ingalls' distinction between the *vision*—which has persisted, and which has attracted and bound individuals to the project over three decades—and the *image*—that which can be critiqued, challenged, and evolved. The actual and *active* relationship between these two entities is not well accounted for in a materialist sociology. Something more is needed to do justice to the cultural history of these systems.

Drawing Things Together

I would like to offer a summing up of the story so far. What is required at this point, after a brief recap of the ground that has been covered, is a refocusing on the larger issues: of the *telos* of this framing, and of its relevance to contemporary society and culture and, of course, to education.

Where we've Been

In Chapter 2, I positioned this study as a kind of *computer criticism*, after Papert (1987), conducted from a place partly inside and partly outside of the tradition it is considering. I have set this work up as *cultural history*, with the emphasis on the historically embedded meaning-making that computing (like anything else) comprises. That is, this is not a technical history, tracing the development of technical artifacts like computers or software programs; nor is it a biography or history of the institutions "behind" particular technologies. Rather, my intent has been to trace the development of the meaning of a technocultural artifact: the Dynabook, itself partly an idealized vision and partly a series of actual technologies. In tracing what the Dynabook or personal computing or technology *means* in changing historical contexts, I must necessarily begin with what it means to me, and this has required that I reconcile my own history with the history presented here.

In Chapter 3, I outlined a theoretical framing of technology as *translation*—both symbolic and material—in order to foreground its constructedness, its contingency, and its essentially political nature. In my discussion of the semiotics of texts and machines, I have attempted to better frame the dynamics of layering and abstraction in technocultural assemblages, drawing on the actor-network theorizing of Latour and Callon, but not, I think confining myself to this framework. Finally, I have introduced the theme of *simulation* in order to highlight the metaphorical function of information technology and thus position us as actors and interpreters.

In Chapter 4, I sketched the key components of the vision of personal and educational computing developed by Alan Kay over the past three and a half decades. This vision began with an insight regarding children's use of computers inspired by the early work of Seymour Papert. Kay's exploration of this area led him to articulate a novel (and still fairly unique) approach to systems design, culminating in the development of a new paradigm of software development: object-oriented programming. Further, Kay and his research team elaborated an application of the developmental psychology of Jerome Bruner in such a way that all three of Bruner's "mentalities"—enactive, iconic, and symbolic—can be employed in human-computer interaction; this work laid the foundations for our contemporary graphical user interfaces and metaphor of "direct manipulation."

Kay's vision of personal computing involves a substantial connection to the history of communications: building on the idea that the printing revolution in early modern Europe saw the rise of a new mode of discourse—structured argumentation. This mode made it possible to articulate ideas not possible in traditional narrative forms, and this shift in discursive practices gave rise to modern science. Newton's arguments on the laws of motion, for instance, are not easily representable in narrative, but rather take the form of sequences of logical assertions—Bruner (1996, p. 89) identifies this mode as *explanation*, distinct from *interpretation*. Kay's contribution begins with the observation that digital computers provide the means for yet another, newer mode of expression: the simulation and modeling of complex systems. What discursive possibilities does this new modality open up, and *for whom*? Kay argues that this latter communications revolution should in the first place be in the hands of children. What we are left with is a sketch of a possible new *literacy*; not "computer literacy" as an alternative to book literacy, but *systems literacy*—the realm of powerful ideas in a world in which complex systems modelling is possible and indeed commonplace, even among children. Kay's fundamental and sustained admonition is that this literacy is the task and responsibility of education in the 21st century. The *Dynabook* vision presents a particular conception of what such a literacy would look like—in a liberal, individualist, decentralized, and democratic key.

In Chapter 5, I moved from a theoretical treatment of Kay's vision to a consideration of the actual unfolding of his research work, loosely in line with Latour's method of "following the actors." Here, I traced the development and evolution of the Smalltalk environment Kay designed in the early 1970s as the platform for his Dynabook vision to the point where it intersects with (a) industrial software engineering practices, and (b) the advent and popularization of the microcomputer in the late 1970s and early 1980s. I outlined here the various *translations* of Smalltalk from an instrument for researching children's "personal dynamic media" to its derivations and incarnations in the computer *industry*. Accompanying the evolution of this particular artifact (or family of artifacts) are a number of related translations—of the role of "users," of the public perception of computing, and of the role of (mass-) market forces.

Kay's work in the 1980s, working within the paradigm of mass-market microcomputers, led him to a deeper focus on the specifics of children's interaction with systems modeling tools, via a long-term research programme at a magnet school in Los Angeles. But concurrent with this in-depth research, the historical trajectory of the personal computer led increasingly away from Kay's ideal of "personal dynamic media" and toward pre-packaged, commoditized, black boxes. Interestingly, nearly concurrent with Kay's behind-the-scenes research in a single Los Angeles school was Seymour Papert's endeavour to popularize and market the Logo system internationally. Logo's trajectory through these years could easily be the subject of another book-length treatment; suffice to say that while Papert and the Logo project enjoyed considerable popularity in the early years of the microcomputer revolution, by the 1990s, the shift in popular conceptions of computing and its significance meant that the ideas being advanced by both Papert and Kay—and thus the Dynabook—were farther than ever from the mainstream.

In Chapter 6, I examined the computing cultures in ascendance in the 1990s, when the Internet and World-Wide Web were coming to prominence. The "powerful ideas" tradition to which Papert and Kay had been central was in retreat, and in its place was a growing conception of computers and the Internet as a kind of mass medium akin to television or

print in which a end users, and, by extension, "learners" play a mostly spectatorial or consumer role. However, at the same time, an older tradition—descended from the 1970s Unix culture—was also gaining momentum, due to the importance of open systems and open standards in the infrastructure of the Internet. The tension between these two trajectories have defined much of popular computing culture in the 1990s and early 2000s. "Educational" computing in this period must be evaluated with these currents in mind: both the vendor-centric notion of commodity software and the Unix-derived open systems culture are in play, often in ironic combinations. But despite the ideological allure and sheer momentum of the Free and Open Source Software (FOSS) movement, I have argued that we should be wary of it as a educational force; for all its overtly democratic rhetoric and anti-corporatist politics, its conceit and stubborn exclusivity (recall the analogy with Latin) makes it a questionable ideal, especially when we contrast it with Kay's child-centric ideal of "personal dynamic media." To repeat the question I posed in Chapter 6, what if we were to be presented with an equally venerable computing tradition, sharing most of the virtues of openness, sharing, simplicity, and modularity inherent in Unix/FOSS culture, but one which was actually designed with children and education in mind?

Clearly, the mere existence of a supposedly better alternative does nothing to ensure or even advance its popularity; Latour and Callon's network model of sociotechnical systems sheds ample light on the dynamics inherent here, and toward the end of Chapter 6 I have included some analysis of how these dynamics seem to play out within computing cultures themselves.

This leaves us with the focus for Chapter 7, in which I have sketched the general features of the contemporary candidate for such a computing artifact: simple, open, shared, modular, and based on a rich and venerable cultural tradition, and yet conceived with children and education in mind from the start. This artifact is *Squeak*, an open-source, Internet-era reincarnation of Smalltalk, born in a curious "back to the future" *topos*[1] in

1. I use the word *topos* here in the sense elaborated by media historian Erkki Huhtamo (1995): cultural-practical assemblages which recur or may be (intentionally or unintentionally) "re-activated" in new technological contexts.

which the core of Kay's development team from the 1970s were seemingly able to re-focus on the agenda set in those early years, but in a world now populated by ubiquitous personal computers, widespread Internet connectivity, and an established culture of shared software development. On this face, Squeak seems to be the best of all possible worlds.

And yet, as the details related in Chapter 7 reveal, Squeak's now nearly decade-long life has been mostly marked by a struggle to define its existence, in ways that interestingly parallel and yet contrast with the ontologically indeterminate *Aramis* of Latour's technocultural whodunnit. This is not to say that Squeak's existence as a piece of software available on the Internet is in question; rather, the struggle is over Squeak's ambitions. Does Squeak represent the coming of the Dynabook, or is it just an open-source Smalltalk implementation, of interest only to a small cadre of programmers? Is Squeak really an "idea processor for kids of all ages" or is it just another authoring environment? Is Squeak an important and revolutionary contribution to educational computing, or is it, as a recently published taxonomy of novice programming environments (Kelleher & Pausch 2005) presents it, merely one of dozens and dozens of kicks at this particular can. On the face of it, Squeak's claim to the loftier of each of these alternatives is questionable. Bluntly put, Squeak's user communities are small and fragmented; its out-of-the-box user interface is baroque; compared with popular computing environments like Unix and MS Windows, it comes across as somewhat alien and peculiar; it is decidely *not* foolproof (as with any serious development environment, Squeak offers more than enough "rope to hang yourself with"); and there is little common understanding of just what Squeak is *for* (Smalltalk development? multimedia presentation? Etoys simulations? user-interface research? web application development?)

And yet, at the same time, in some very real and overt ways, Squeak is a vastly better computing environment than anything anyone has come up with: in terms of sheer flexibility it is unsurpassed; as an exercise in software portability and device independence it is probably without peer; it has the facility to enable children and novice users to work with sophisticated systems modelling very quickly and easily; it is a latter day version of one of

the simplest and most elegant programming languages ever developed; and it is completely and unrestrictedly open, from top to bottom.

Despite its virtues, the extent of Squeak's *network* (in Latour and Callon's sense) is extremely limited—compared with the vast community of developers working in the Unix/FOSS tradition, or, alternatively, the staggering market-driven scope of Microsoft's version of personal computing. And yet, most interestingly, despite the weaknesses and strengths I've listed here, Alan Kay and his team seem almost uninterested in the fate of Squeak in the marketplace of ideas.[2] Kay is explicit about this: Squeak is only a vehicle to get to the next place; it is decidedly not the destination, not the end itself. This echoes the trajectory of Smalltalk at Xerox PARC in the 1970s, a cycle of re-invention and self-transcendance. With Smalltalk-80 and its release to the wider world, Kay lamented, Smalltalk stopped being reinvented every few years and instead became cemented as a black box, its own "inflexible religion." Squeak, in yet another back-to-the-future move, is seemingly a return to this aesthetic; born as a Smalltalk-80 implementation, its core developers have not shied away from altering nearly every aspect of the software; from its user-interface framework in 1997 to a 2005 movement toward an architecture called "traits," which threatens to replace even the class-and-instance model core to Smalltalk.[3] So, to watch the contemporary trajectory of Squeak as it spawns new projects like Croquet is not surprising in this light. With each iteration, new facets of the Dynabook vision are prototyped, realized, perfected, evaluated, and sometimes abandoned. Squeak, as I pointed out in Chapter 7, is not the Dynabook. To get an idea of what the Dynabook really is, we are required to take a longer view.

2. As Kim Rose points out (personal communication, Oct 2004), there is no marketing effort behind Squeak, nor are there plans for one.
3. See the *squeak-dev* mailing list in 2005 for ongoing discussion of this possible development.

Dynabook: Artifact or Idea?

What are we to make of such a case? Is the Dynabook real, or not? Smalltalk is *real*, but it isn't the Dynabook. Squeak is *real*, but it isn't the Dynabook either. Nor is my laptop computer, which bears more than a subtle resemblance to Kay's early mock-ups. Like Xerox' *Alto* minicomputers circa 1973, these are all "interim dynabooks" or facets thereof. So, when can we drop the "interim" qualifier? Surely at some point, a lightwight, network-connected portable computer running some kind of open and user-modifiable object- (or perhaps message-) oriented software—in the hands of children—must add up to the real thing. Or, alternatively, perhaps this is the tragic story of a failed vision, an artifact who has failed to *be*, like Aramis.

The trouble with this sort of framing is a precisely the technocentric thinking that Papert warned against in his 1987 call for "computer criticism." The point of casting this as *cultural* history is that we cannot evaluate the Dynabook as one or other assemblage of hardware and software. Now, here is an obvious parallel with Latour's brand of techno-sociology; Latour would tell us that any technological system must be viewed as an assemblage of social, technical, human, nonhuman, physical, and political actants. Latour's treatment of Aramis is precisely the elaboration of the multitude of networks which constitute that ill-fated project. But here I feel I must depart somewhat from Latour's otherwise powerful framework.

The fate of Aramis, Latour tells us, lay in the unresolved contestation among the many actors:

> The war of interpretations continues for Aramis; there are only perspectives, but these are not brought to bear on anything stable, since no perspective has been able to stabilize the state of things to its own profit. (Latour 1996, p. 79)

Latour warns us that "There is no such thing as the essence of a project. Only finished projects have an essence" (p. 48). But perhaps we must make a distinction between technological paradigms.[4] The 'object' of Latour's analysis is a large-scale, immensely expensive,

technosocial assemblage, composed of vast quanitites of steel, rubber, concrete, workers, passengers, governmental agencies, funding bodies, magnetic couplers, navigational computers, and so on. It is all Aramis can do to struggle to *come into* existence; indeed, the sheer inertia Aramis must overcome is enormous and multifaceted: physical, technical, political, budgetary. The chief requirement is co-ordination, the establishment of a common frame of reference, and indeed, Latour's account is one of the dynamics of the work of establishing (by means of alignment, displacement, translation, and the rest of Latour's conceptual repertoire) a common trajectory defining a single technosocial artifact.

The Dynabook, in contrast, almost *begins* its life with a common frame of reference: a vision shared by many contributors and conceptually linking several instantiations. Where Aramis' essence follows (or fails to follow) its existence in implementation—and where the *details of implementation* (in terms of design, engineering, funding, political will, etc.) are the primary site of struggle in Latour's account, the various implementation(s) of the Dynabook are almost *emergent* from the interplay of ideas and technocultural contexts. The implementation itself is, as Kay's repeated appeal for Smalltalk's self-transcendence points out, *relatively unimportant*—as it was with McCarthy's Lisp, which first saw embodiment on paper, and which later—perhaps *less* profoundly—became a family of software programs. But if, as this suggests, the actual implementation is not important, what are we left with? Without the concretion of black boxes which may be closed, objectified, reified, what are techno-social networks to be composed of? If we take away the necessity of actual technical artifacts, are we not left with human beings, with pure society? Do we not find ourselves back with social constructivism—*a la* Bijker, Hughes, & Pinch's *Social Construction of Technological Systems* (1987)—a stance from which Latour spent the better part of two decades attempting to distance himself.

I am not prepared to return to a naïve social constructivist position—far from it. The sticking point is, I think, the stubborn *materialism* inherent in actor-network theory.[5] I

4. I use the term "paradigm" here in its mundane, pre-Kuhnian sense of a defining example.
5. John Law (1992) even goes so far as to coin the term "relational materialism" in the context of actor-network theory.

refer here not to "materialism" in the sense of belief or nonbelief in an "external reality," but rather the materialism which is shot through contemporary sociology, inherited originally from Marx and more recently from Garfinkel, and which underlies most of contemporary science and technology studies. I am most certainly not prepared here to write an idealist critique of materialism in social science, but I do want to pose, as a specific challenge, the concept of a "powerful idea" and how we can account for it in a materialist epistemology. In most sociological accounts *ideas* must either take the form of embodied representations, like inscriptions (see Latour & Woolgar 1986) or documents (see Brown & Duguid 1996), or, alternatively, as "discourse effects" as with Foucauldian method (within STS, we would say "network effects")—that is, somehow emergent from the materialist base. Latour's notion of a "parliament of things" and his emphasis on including the nonhumans with the humans is huge improvement over the social construction of technology, but its materialist underpinnings mean it is still a blunt instrument for considering the broader cultural dynamics of ideas themselves. Something like Gadamer's phenomenological hermeneutics, with its emphasis on the *experience* of interpretation and meaning-making, is required to do this work.

I do not mean to suggest here some non-mediated essence for ideas; my starting point in this discussion has been the foundational mediatedness of all human experience. What I am calling for here, then, is a broadening of what counts as mediation. An *idea* like the Dynabook—like more powerful ideas such as the theory of evolution; general relativity; democracy; or complex systems modelling—exists by way of its *currency* in "discursive communities" or "communities of practice"—its mediation is by way of artifactual embodiment, textual and documentary inscription, individual utterances, and probably some (as yet) elusive cognitive representation as well. My point here is that this laundry list of media does not do a very good job of shedding light on what "currency" might mean with respect to ideas.[6]

Let me wrap up this methodological aporia with the following suggestion: theories of technoscience have so far been successful in explaining the *concretion* of theories,

constructs, and systems—the accounts made in STS research of the laboratory practice, systems of classification, microbe theories, bicycles, jet planes, and rapid transit systems. But these forms are not the whole of technoscience. They may well be the tip of the iceberg; the visible, empirically studiable 10% of what actually comprises actual practice. And, in a moment that recalls the fish's difficulty with the concept of water, it may be that the *cultural* aspect of technoscience is still woefully underrepresented—the aesthetic dimension; the virtues, values, and vices; the drives and forces that actually move technoscience along; these are mostly absent from most STS accounts.

Cultural history is not Biography

Having now done uncertain violence to accepted sociological method, cultural materialism, and, for that matter, media studies, I must address a further possible question, one which will, at this point have no doubt been raised in my readers' minds: that the Dynabook is an idea which exists in the head of an individual man, that of Alan Kay, and that its social or cultural existence is dependent upon his continual—stubborn, even—reiteration of it. This possibility is also uncomfortably linked with the notion that what I have been writing here is nothing but a "great man" theory of history. Admittedly, there is much in Kay's story that lends itself to this kind of treatment.

I must try to defend against this charge, however. Despite Kay and his philosophy being the primary narrative vehicle for my account, and despite his inescapable centrality to the Dynabook, this story is not his. In the very first place, the Dynabook traces itself to Seymour Papert, whose early (1968) and continued relationship with Kay and his work is absolutely key to this story. Second, the role of Kay's colleagues—Dan Ingalls, Ted Kaehler, Adele Goldberg, Diana Merry—is enormous. It is perhaps a pitfall of this and other accounts that Ingalls may appear as the implementor of Kay's ideas. In fact, Ingalls is no less a conceptual innovator, and the ongoing tension between Kay's and Ingalls' vision is one of the central

6. Richard Dawkin's popular notion of a "meme" as the unit of cultural transmission is no doubt raised here, though I find this concept lacking on a number of counts; suffice it to say that "transmission" is a poor way to talk about the performative practice of culture. That said, at least Dawkins' notion at least *attempts* to account for the spread and currency of ideas within communities.

dynamics here. Ingalls' 1981 article, "Design Principles Behind Smalltalk" is every bit a manifesto on the order of Kay's and Goldberg's writings; here is a text which has as much or more to say to the question of why we should care about digital technology as it does to the particular architecture of object-oriented systems.

But beyond merely pointing out that the Dynabook story involves a larger group of people than Kay, the truly compelling point of the story is how this vision is shared by various participants, *in the absence of any extant, concrete instantiation*. Of the team at Xerox PARC, Kay recalled,

> I needed a group because I had finally realized that I did not have all of the temperaments required to completely finish an idea. I called it the Learning Research Group (LRG) to be as vague as possible about our charter. I only hired people that got stars in their eyes when they heard about the notebook computer idea. I didn't like meetings: didn't believe brainstorming could substitute for cool sustained thought. When anyone asked me what to do, and I didn't have a strong idea, I would point at the notebook model and say, "Advance that." LRG members developed a very close relationship with each other—as Dan Ingalls was to say later: "... the rest has enfolded through the love and energy of the whole Learning Research Group." A lot of daytime was spent outside of PARC, playing tennis, bike riding, drinking beer, eating chinese food, and constantly talking about the Dynabook and its potential to amplify human reach and bring new ways of thinking to a faltering civilization that desperately needed it (that kind of goal was common in California in the aftermath of the sixties). (Kay 1996*a*, p. 527)

This is a story of a cultural phenomenon, not a personal agenda. It crystallized in Kay's mind, or perhaps in his lab, in the late 1960s and early 1970s (and, as I have noted, this makes the job of the historian much easier) but since that time it has been the site of the practice and participation of scores of people. Interestingly, despite the myriad translations—some of which I have traced in the preceding chapters—the core vision remains today, quite recognizable, still a centre of gravity for those who share it. This is, again, an aspect poorly accounted for in actor-network theory or the sociology of translation, which states that for a technosocial system to be successful and achieve currency, it must necce-

sarily undergo various translations which move it from an isolated, perhaps "pure" conception to a strongly reinforced, network of aligned forces. This kind of account serves well to explain the trajectory of Smalltalk over three decades, and even the relative frailty of Squeak, but it speaks nothing to the cohesion of the idea, of the vision, over thirty-plus years.

Back from the Future

Yet another accounting I must address here is a much bleaker version of the story: simply that the Dynabook does not exist, nor is it likely to. In this version, I need to go to no pains to account for the coherence of the vision over time, nor the respective translations and survivals of facets of the Dynabook, because there aren't any. In this version, Kay's early vision was quickly turned into other things—object-oriented programming, the mass-market PC—and any latter day reminiscence on the romantic vision driving the story is but the nostalgia of those whose day is, frankly, past. It is not difficult to find commentaries which take this basic stance (e.g. Dugan 1994; Bardini & Horvath 1995). I do not, however, find these analyses comprehensive or convincing, mostly because I reject the assumed 'irreversibility' and complacency in such accounts.

Let me engage in a brief thought experiment: suppose the year is 2016; by this point, in Canada, what used to the called the Internet is now taken for granted to the point of being almost invisible; similarly, computing devices have been miniaturized and proliferated to the point where people do not even talk about "computers" anymore. Within such a fine-grained digitally mediated environment, software programs have ceased to be distinct things unto themselves; rather, software is a vast conversation of distributed message-passing objects; application software has given way to network-hosted services. Schoolkids commonly use computing technology in their classes—usually in the construction and modification of virtual reality models of processes and systems, some taken from the natural world (as in life sciences) but many more from the digital sphere. In this future, an episode like the one in Kay's 1972 paper—children sitting on a grassy lawn collaboratively playing

with the workings of a video game—is commonplace; the difference is that instead of each child holding a personal computer, children tote around small notepad-like devices which serve as user interface gadgets, connecting them to a global computing grid.

This scenario, as far as I can see, and barring some kind of global catastrophe, is an entirely reasonable one given current trends in digital technology.[7] If such a scenario were indeed to come to pass, historians would be in a position to trace the roots of such techno-social assemblages. And in doing so, some of that historiography would no doubt intersect with much of what I have written in the present document. The very real possibility of such histories being written in the future makes any present claim that the Dynabook is simply a failed vision (as of 2006) untenable. If in, 2016, we witness a very different kind of digital world, we will then perhaps be able to say that the Dynabook was a failed vision which never existed. But for now, in 2006, such a scenario is still *as likely as not*, and as such, it would not be legitimate nor even reasonable to foreclose on its historical potential.

This raises an interesting question—or nest of questions—about the future of the Dynabook vision. The simplest question is whether, as time goes on, and the partial and contingent implementations of various facets of the Dynabook multiply, the Dynabook vision becomes diluted; that, by extension, in ten years time will this historical thread even be identifyable in the face of its sheer plurality? A second simple question—a less interesting one, I believe—is whether the Dynabook 'project' will simply run out of steam or be substantially translated in ways far more serious than what we have seen to date. What if Alan Kay himself—already in his 60s—simply retires to his pipe organ and chamber music, and leaves a younger generation to maintain the vision? Is there enough coherence in the Dynabook *idea* (and its manifestations) that it would survive the absenting of the original cast—Kay, Ingalls, Kaehler, for instance?

It is difficult to mount answers to these questions that go beyond mere speculation. A few comments about future directions are warranted, however:

7. While not "utopian," I would say this scenario is wholeheartedly optimistic. I can easily come up with equally feasible versions featuring nasty corporate domination—more on that later.

- Core Squeak developers, led by Andreas Raab, are currently working on a successor user-interface framework called *Tweak*. Tweak is aimed at what Kay has called the "omniuser"—that is, novice and nonprofessional users in general, as opposed to just children. However, the entire Etoys system appears to be being re-implemented in Tweak. This may have the effect of expanding the immediate applicability and user-base of Squeak beyond the Etoys-oriented school community and the programmer-oriented Smalltalk community to include a "middle" tier.
- The *Croquet* environment, a networked, multiparticipant, 3D environment is picking up momentum well beyond Kay's own team, at a number of US universities interested in it as a higher-educational platform. Its appeal to a generation raised on first-person 3D video games is clear, as well. As an Internet application, Croquet is forward-looking to the extent that it not only substantially leapfrogs Squeak's existing desktop-client environment but perhaps the World-Wide Web itself. Indeed, the Croquet project received attention in the blogosphere in Nov, 2005, billed as a "Web 3.0" application—a tongue-in-cheek reference to current hype about an emerging "Web 2.0" paradigm.
- Mainstream computing—and, by extension, the kinds of systems ultimately found in schools and other educational contexts—shows evidence of a gradual, if fragmented, trend toward more Smalltalk-like systems: newer programming languages like Python[8] and Ruby have established better compromises between the Unix tradition and the Smalltalk aesthetic, and in recent years there is substantial discussion of the merits of "dynamic languages" over Java (Tate 2005). It may be that elements of the Dynabook vision may gain currency *despite* Squeak's or Smalltalk's relative success.[9]

8. The Python community has a dedicated educational thread within it, embodied on the edu-sig@python.org mailing list. In spring 2006, a "summit" of sorts was hosted by the South African Shuttleworth Foundation, bringing together representatives from the python community as well as Alan Kay, Kim Rose, and other Squeak people. The possibility of a python-based Etoys-like environment was discussed on mailing lists on both sides of this cultural divide. See http://wiki.tsf.org.za/shuttleworthfoundationwiki/Day_one

- A recent development I find myself at a loss to quite/yet evaluate, *pro* or *con*: in November 2005, at the World Summit on the Information Society in Tunisia, UN Secretary-General Kofi Annan presided over the unveiling of the prototype for MIT's "One Laptop Per Child" project, an endeavour of astonishing ambitions led by MIT Media Lab director Nicholas Negroponte. The plan is to produce a $100, handcrank-powered notebook computer in the tens of millions and distribute them to schoolchildren in developing nations. The laptops run open-source software (based on Linux), feature wireless connectivity and software for creating (decentralized) mesh networks. The laptops will reportedly have Squeak installed; Alan Kay and Seymour Papert are among the 14 principals behind the project.[10]

Whatever we make of these individual items, it does seem clear that there is plenty of energy and even momentum at work here, despite the fact that it is hard to tell exactly in what direction it is going.

Who Cares About the Dynabook?

Theoretical issues and speculation aside, there remains the question of the relevance of this story to education and educators. I have, I believe, outlined a story of a particular innovation's gradual movement away from the mainstream; however brilliant Kay's ideas may have been in the 1970s, there is no denying that in terms of actual practice, they must appear marginal and perhaps even quaint in the face of large scale institutional e-learning, distance education, virtual schooling, computer-mediated communications, standardized assessment, classroom podcasting, and what-have-you. Here, as elsewhere in this analysis, the

9. As of early 2006, Dan Ingalls is employed at Sun Microsystems and has been working on a Java-based implementation of the Squeak virtual machine.

10. A press release dated Dec 13, 2005 states, "The One Laptop per Child (OLPC) board of directors today announced that Quanta Computer Inc. of Taiwan was chosen as the original design manufacturer (ODM) for the $100 laptop project. [...] In announcing the selection of Quanta, OLPC Chairman Nicholas Negroponte said, 'Any previous doubt that a very-low-cost laptop could be made for education in the developing world has just gone away.' Quanta has agreed to devote significant engineering resources from the Quanta Research Institute (QRI) in Q1 and Q2, 2006, with a target of bringing the product to market in Q4. The launch of 5–15 million units will be both in large-scale pilot projects in seven culturally diverse countries (China, India, Brazil, Argentina, Egypt, Nigeria, and Thailand), with one million units in each of these countries." See http://laptop.media.mit.edu/news.html

contemporary *meaning* of educational computing is wrapped up tightly with that of personal computing. What the latter has come to represent has everything to do with the former. Kay's three decades of work remain in ambiguous relation to both.

Kay's relevance to education

I will begin with a frank observation: almost no one I have talked to in the educational establishment—teachers, administrators, professors, graduate students—has any clue who Alan Kay is. Most have heard of Seymour Papert, and can even give a basic outline of Papert's project. But Kay and his work remain almost entirely unknown. This we must contrast with Kay's profile within computer science, where he is most certainly known—in 2004 Kay was the recipient of the Turing Award Association of Computing Machinery's highest honour—and his influence (and critique) widely felt. Of course this is not surprising; Kay's professional career has been spent among computer scientists, and his practical association with schools and education limited to a handful of schools. Papert, in contrast, was something of a public celebrity in the 1980s, and the author of an enormously popular book (Papert 1980). Nevertheless, and without wanting to diminish Papert's contributions at all, it must be pointed out that Kay's actual influence on personal and educational computing is vastly greater. The dominant desktop paradigm and our contemporary understanding of "authoring" software had their origins in Kay's lab. If we consider the small but significant impact of the object-oriented paradigm in information technology and computer science curriculum, one could say that Kay's influence is felt in a third area as well. The first point that the present study seeks to establish is that all these now-commonplace motifs had their origins in the pursuit of an educational platform, one seriously informed by the educational psychology of Piaget, Bruner, and Montessori. Desktop computing and multimedia authoring were not conceived as tools for office workers or even media professionals—they were in the first place tools for children and elements of a vision of literacy for the 21st century.

That this emphasis is lost on us now is perhaps easily attributed to historical distance; the world is a different place than it was in 1972—and even the world of 1972 was not well

represented by the San Francisco Bay Area of the day.[11] This has not been a study of the emergence of the personal computer; that topic is much larger than any of the several books already published on the topic, and which generally give short shrift to the role played by Kay's group at Xerox. Rather, I have tried here to stay focused on the educational vision, and, for simplicity's sake to treat the emergence of personal computing as a function of the that. Complicating this is the serious divergence and disconnect between the discourse of computer science and that of education, even educational technology. Those who are capable of speaking fluently in both realms are few and far between, a fact easily and often underestimated by both the readers and authors of educational technology texts. To underscore this, consider that in the 1970s, a number of the *leading* thinkers in the field of computer science (at places like Xerox PARC, MIT, and Stanford) had turned their attentions to the needs and concerns of education. How different this is today! That the 2004 Turing Lecture was delivered by someone whom I would in all seriousness consider—and I hope I have made this apparent in the preceding pages—an educational theorist is a complete anomaly.

Why should we care about this? Why should it be important that computer scientists have their minds—god forbid their hands—on education? There are several possible answers to this question.

One possible answer is that the education system—in the United States, for instance—is seen to be badly in need of reform, and the introduction of computers is seen as a possible vector for reform. Papert has suggested as much in several of his writings (though I would not go so far as to say this has been his primary motivation), as has Kay (1972, p. 1).

A second possible answer has to do with improving the administrative and logistical efficiency of educational systems: this is clearly the aim of learning management systems, computer-driven assessment systems, and a host to drives toward standardization. But little of this area can be of serious interest to computer science as an intellectual discipline.

11. John Markoff's (2005) *What the Dormouse Said: The Untold Story of How the Sixties Counterculture Shaped the Personal Computer Industry* sheds much light on the cultural zeitgeist at labs like Xerox PARC in the early 1970s.

A third, and more interesting answer has to do with the now half century-old tradition of cognitive science. This is the realm in which the early researchers in artificial intelligence were operating (including John McCarthy and Seymour Papert). In this framing, research into the modeling of the mind and research into how learning takes place are seen as facets of a single, larger pursuit. A simple example of this concept is the now well-known concept of "multiple intelligences" advanced by Howard Gardner (1993), and its application to the design of multimedia curriculum resources.

But there is another possible answer to the question of why computer science is important to education, and that is the one having to do with "powerful ideas"—this is clearly the primary driver for both Papert's and Kay's projects. In this framing, computer science itself is not the important thing, but what it makes possible: the study and engagement with complex or dynamic systems—and it is this latter issue which is of key importance to education. Indeed, Kay has aggressively questioned the very existence of a "computer science" as a discipline unto itself (attacking the notion of "software engineering" in the same breath—see Kay 2000*b*). But a "systems science," of which computing is legitimately a part, in the same way that print literacy is an integral part of modern science, is surely of real importance to education. Note that this is not the same knee-jerk, means-ends argument that has been repeated over and over again in critiques of computers in education; that is, this is not the same as saying that educational ends must take precedence over technological means. Kay has read his Marshall McLuhan, and we would do well to recall him here, and not suppose that the "message" is somehow independent of the medium. But it does not mean either that the medium is the important thing in and of itself. Kay says it best:

> The reason, therefore, that many of us want children to understand computing deeply and fluently is that like literature, mathematics, science, music, and art, it carries special ways of thinking about situations that in contrast with other knowledge and other ways of thinking critically boost our ability to understand our world. (Kay 1996*a*, p. 548)

This is the point of the Dynabook, and the kind of literacy it suggests: like the printed book before it, the computer has the possibility to "carry special ways of thinking" that provide access to the most powerful ideas of our age. The underlying political question here is who will have access to those ideas and their application?

Education and Powerful Ideas

What are "powerful ideas"? BJ Allen-Conn and Kim Rose, following Papert, say they are "intellectual tools" (2003, p. *v*). What makes an idea powerful? Its connectedness to other ideas, and how far they can take you; using Latour's vocabulary, we would say that "powerful" ideas are those with extensive networks. Kay's repeated critique of education and schooling is that most powerful ideas have been either deliberately or inadvertently evacuated from curriculum.

> Much of "school math" is not mathematics at all but attempts to train children in various kinds of calculation using patterns and recipes. Mathematics is actually about representing and thinking clearly about ideas. Science goes further: to try to come up with plausible ideas about the universe that are worthwhile thinking clearly about. (Kay 2004*b*)

It is not, in this day and age, terribly difficult to come up with examples of how the school system is failing, and it is not my agenda to do so here. Rather, I want to underscore a simple idea or principle: that powerful ideas of the kind described here—whether they be the geometry of triangles, the theory of evolution, the idea of liberal democracy, or the dynamic modelling of population dynamics—*are precisely the responsibility of education*. Powerful ideas so defined imply particular ideas of literacy. To be fluent with trigonometry, one must be literate with respect to diagrams, lines, points, angles, algebra, and so on—to read them and to construct them oneself. To fully grasp the idea of liberal democracy, one must be able to read, analyse, and critique a number of document forms, and to be able to construct such arguments oneself. The kinds of powerful ideas emergent in the digital age—simulations, systems dynamics, multimedia, collaborative works, and so on—mean one must become conversant with the genres and methods of elaborating such constructs, and to be able to

create and manipulate them oneself (see Lemke 2001). Are these not the special province of education? Is not the *ideal* of public education, mass education, that these ideas be accessible to all, and that the very structure and health of democratic society depends upon their reasonably widespread fluency (Dewey 1916; Kellner 2006)? And furthermore, as my emphasis on the ability of the literate to themselves *construct* and actively *participate in* each of these forms hopefully suggests, is it not the responsibility of education to nurture not the *givenness* of these ideas, but our own role in their ongoing *construction and definition*. Is this not what democractic citizenship—as framed by Dewey—is all about?

The Politics of Software Revisited

Alan Kay's conception of powerful ideas is focused primarily on science and math. But it important to not limit the scope of the argument to particular curricular areas. Indeed, Kay has argued that education in the United States has not done justice to much of the print culture of the past three or four centuries. Mathematics and science are but emblematic of this larger interest in what literacy really means.

> Most of the important ideas in our civilization are not on the Internet yet, but they are already available in free public libraries. The haves and have-nots during these coming years will not be the people who do or do not have access to the Internet. The real gap will be between those who are discerning about the information they access and those who are not—between those who use technology to help them understand powerful ideas and those who do not. (Kay 1997, p. 19)

I wish at this point to move this discussion beyond Kay's own framing, and to take a page from Richard Stallman, who in the 1980s founded the Free Software Foundation and who arguably remains the most important thinker of the Free and Open Source Software (FOSS) movement.[12] In an interview given in 2001, Stallman commented,

12. Stallman would balk at being identified with "open source" software, since he has remained vehement that the issue is one of "freedom." However, I use the more inclusive term here to refer to the cultural movement as a whole.

> I've dedicated 17 years of my life to working on free software and allied issues. I didn't do this because I think it's the most important political issue in the world. I did it because it was the area where I saw I had to use my skills to do a lot of good. But what's happened is that the general issues of politics have evolved, and the biggest political issue in the world today is resisting the tendency to give business power over the public and governments. I see free software and the allied questions for other kinds of information that I've been discussing today as one part of that major issue. (Stallman 2001)

Stallman's focus here is very different from Kay's. He is not talking about education, and he is not concerned with the status of mathematics or science *per se*. What Stallman is alluding to is the issue of ownership of knowledge, and from where he stands, this issue is not an academic or philosophical one, but a very real, concrete problem. As our daily work and lives are increasingly conducted and mediated via digital media—and I don't think anyone who has not been asleep for the past decade would downplay this—*software* is more and more the 'material' substrate of our culture. Stallman led a charge, beginning in a very marginal way in the 1980s and growing to enormous importance in the late 1990s, about the relationship between software (both development and distribution) and market capitalism. Stallman was appalled at the extent to which his work as a programmer came to be at the behest of and within the strictures of corporate capitalism. His response to what he defined as a "stark moral choice" was to found the Free Software Foundation and establish the GNU project in order to construct a computing platform on terms not governed by the increasingly mainstream terms of copyright exploitation and restrictive licensing. At the time, the only people who cared about this were Unix programmers. But, with the popular rise of the Internet and World-Wide Web in the 1990s—and the translation of Stallman's GNU project into the GNU/Linux project and the wider FOSS movement, Stallman's concerns and his response have intersected with mainstream thinking about information technology.

But, to date, most thinking about Free and Open Source software has focused on the mechanical aspects: that it is possible to develop and distribute software without demanding payment for it; that the quality of the software developed under such conditions is in

some cases of higher quality due to the operation of something like peer review in the development community; the existence of a commonwealth of software code increases the benefit for all. What has so far been downplayed—indeed the emergence of the "Open Source" branding is precisely such a move—is the political element of Stallman's argument; that the important thing is not the price of software, nor the formula for quality, nor the efficiency of the development community. What is truly important, Stallman argues, is the *freedom* we enjoy in being able to develop, use, share, distribute, and build upon the fruits of one another's efforts. Stallman's is a political argument first, not an engineering or a marketing argument.

How do we as a society ensure freedom? The mechanical conception of FOSS places the emphasis on a particular application of copyright law; the GNU General Public License, among others, seeks to ensure that software released under its terms remain free in all the above mentioned ways. But the license is not the end of the argument; the license—and the laws it draws upon—are expressions of a particular culture. So the deeper and more difficult question is how do we as a *culture* ensure freedom?

Legal scholar Lawrence Lessig has perhaps tackled this aspect of the question more comprehensively than most. In perhaps his best-known work, a lecture entitled "Free Culture" (2002), Lessig characterizes the current fight over copyright laws—that which Stallman calls "the biggest political issue in the world today"—as the struggle for who owns and controls creative expression in our society. In his lecture, Lessig repeats this "refrain":

- Creativity and innovation always builds on the past.
- The past always tries to control the creativity that builds upon it.
- Free societies enable the future by limiting this power of the past.
- Ours is less and less a free society. (Lessig 2002*a*)

How does a society ensure freedom? By valuing it, in the first place—and in the second place, by knowing what it means to be free. In a technologically mediated society—and all societies are technologically mediated—this means attending to the dynamics of expression, and the modes of mediation. In the era of print, liberal modernity enshrined the idea of

a "free press" and the extent to which our nations and societies have taken this idea seriously (vs. paying lip service to it) has been an ongoing debate of crucial importance. In the era of print, we are fortunate that we have *at least been aware* of this debate; even when we are most cynical about the reality of a free press. It is at the very least an open political question, and thus a cultural touchstone or reference point to which most "educated" people can allude.

But we are today rapidly altering our media landscape. And as we do so, as we move into unknown waters, we cling to our existing constructs and ideals as though they were lifejackets. We know deep down, however, that the political constructs of a prior age are not likely to be quite apt in a new era, in which our culture operates by different means, in which our thoughts and expressions and interactions are mediated by different systems. The current controversy over copyright law and copyright reform should at least highlight this issue.

If, as I suggested above, in this new digital age, software is more and more the substrate of our culture, the material of its expression, and thus the site of political articulation and struggle, then software freedom is indeed the issue of our time, as both Stallman and Lessig would say. To this I must add; it is not going to be sufficient to ensure freedom in software merely by releasing it under a particular license, written according to the terms of copyright law.[13] What is going to be required is a change in culture: how we understand software, and how we understand freedom.

In Chapter 6, I made a case for why I believe that the FOSS movement itself is inadequate for bringing about such changes at a society-wide level. The FOSS movement has succeeded in changing the cultural context of software for those already in the know, or those who for other reasons are prepared to take the time and effort to understand the issues it addresses. But, by virtue of the intellectual and cultural heritage FOSS draws from, it is unlikely to make headway more generally in our society. To the contrary; in its exclusivity, it actively works against its own mainstreaming. And, at the end of that Chapter, I

13. As Lessig and numerous others point out, copyright law is an artifact of the advent of print; aimed specifically at circumscribing the political power of those who controlled the presses.

suggested that the tradition represented by Kay's work was perhaps a better candidate, on the grounds that here was a paradigm of software development equally free and open, yet much more pedagogically sound, having its roots and first principles firmly rooted in a vision of *personal* computing, articulated in a *peer-to-peer*, *do-it-yourself*, *learner-centric*, *less-is-more* aesthetic. Note that these latter are not 'features' of a software product; they are *virtues* of a computing culture.

Software is political. This is obvious in the mundane sense, as anyone who has ever had to warp their personal practices and expressions into a pre-conceived digital framework can attest. It is also obvious at the level of high-level poltical-economic analyses of the computing industry or the military industrial complex (e.g. Edwards 1996). But it is also true, more importantly, at a more profound and subtle level: at the level of culture—of the construction and co-construction of meaning and significance. What should we as individuals *expect* from digital media? Should we learn to expect it to be malleable, manipulable, combinable, and understandable? Or will we be content to accept pre-packaged, shrinkwrapped capabilities, defined by corporations and marketers?

And where will we learn such things? In school, will our children learn the logic and dynamics of digital media, in the sense I outlined in Chapter 3? Will our children learn that they are born into a mediated world more malleable, recombinable, and translatable than any other age has even dreamt of? Or will they learn in school that sharing is a crime, that Microsoft Office defines the set of possibilities for document communication, and that the limits of cultural expression are basically those set forth by Hollywood? The fact that our education system is woefully underequipped to even begin to answer these questions is nothing new; a generation of critical educators have already elaborated school's impotence and even complicity with structured dominance (e.g. McLaren 1993; Kincheloe & Steinberg 1995; Torres 1998; Apple 2004). I am not about to suggest that the adoption of a technological platform will do anything to address such a large-scale systemic issue, nor to throw my lot in with those who have seen the computer as the "trojan horse" for school reform (Greenberg 2001). I rather appeal to the notion that education is reflective of the culture it

serves, and as such see more promise in the evolution of computing and media cultures online. Even the FOSS movement, for all that I have said in critique of it, at least carries a certain weight, and the political messages emanating from it at least point in a constructive direction.

As for the furtherance of the Dynabook ideal, I hope that by now I have made a case for this being a rather diffuse process, rather than a singular trajectory. I will end with a statement of optimism: that there is enough constructive chaos in the digital world today, enough energetic searching for wisdom and meaning on the part of millions of disorganized (at least by conventional definitions of organization) people and enough potential for the discovery—or better, the construction—of powerful ideas that the virtues embodied in the computing culture of the Dynabook vision are as likely to prevail as the opposite, corporatist alternative, which also boasts formidable resources and momentum. Our challenge, then, as educators, practitioners, and philosophers, is to attend to the business of recognizing, appreciating, criticizing, and articulating the good. My sincere hope is that the preceding work is a contribution to that effort.

Bibliography

Aarseth, E. (1997). *Cybertext: Perspectives on Ergodic Literature.* Baltimore: The Johns Hopkins University Press.

Aarseth, E. (2004). Genre Trouble. *Electronic Book Review, 3*.

Abelson, H. & Sussman, G. J. (1996). *Structure and Interpretation of Computer Programs* (Second ed.). Cambridge: MIT Press.

Agalianos, A., Noss, R., & Whitty, G. (2001). Logo in Mainstream Schools: The Struggle over the Soul of an Educational Innovation. *British Journal of Sociology of Education, 22*(4).

Alexander, C. et. al. (1977). *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press.

Allen-Conn, B. J. & Rose, K. (2003). *Powerful Ideas in the Classroom:: Using Squeak to Enhance Math and Science Learning*. Glendale: Viewpoints Research Institute.

Ambron, S. & Hooper, K. (Eds.). (1990). *Learning with Interactive Multimedia: Developing and Using Multimedia Tools in Education*. Redmond: Microsoft Press.

Apple, M. W. (2004). *Ideology and Curriculum*. (Third ed.). New York: RoutledgeFalmer.

Apple Computer Inc. (1995). *Teaching, Learning, and Technology: A Report on 10 Years of ACOT Research.* Apple Computer Inc. Retrieved October 5, 2006, from http://images.apple.com/education/k12/leadership/acot/pdf/10yr.pdf

Bardini, T. & Horvath, A. T. (1995). The Social Construction of the Personal Computer User. *Journal of Communication, 45*(3).

Baudrillard, J. (1995). *Simulacra and Simulation* (S. F. Glaser Trans.). Ann Arbor: University of Michigan Press.

Bazerman, C. (1998). Emerging Perspectives on the Many Dimensions of Scientific Discourse. In J. Martin & R. Veel (Eds.), *Reading Science: Critical and Functional Perspectives on Discourses of Science* (pp. 15–28). London: Routledge.

Beck, K. (2000). E*xtreme Programming Explained: Embrace Change*. New York: Addison-Wesley.

Berners-Lee, T., Henders, J. & Lassila, O. (2001, May 17). The Semantic Web. *Scientific American.* Retrieved October 5, 2005, from http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

Bernstein, R. J. (1983). *Beyond Objectivism and Relativism: Science, Hermeneutics, and Praxis.* Philadephia: University of Pennsylvania Press.

Bezroukov, N. (1999). Open Source Software Development as a Special Type of Academic Research: Critique of Vulgar Raymondism. *First Monday, 4*(10).

Bhabha, H. (1994). The Commitment to Theory. In *The Location of Culture*. London: Routledge.

Bijker, W. E. & Law, J. (Eds.). (1992). *Shaping Technology/Building Society: Studies in Sociotechnical Change*. Cambridge: MIT Press.

Bijker, W. E., Hughes, T. P., & Pinch, T. J. (Eds.). (1987). *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Cambridge: MIT Press.

Bloor, D. (1999). Anti-Latour. *Studies in the History and Philosophy of Science, 30*(1), 81-112.

Booch, G. (1991). *Object Oriented Design with Applications*. Redwood City: Benjamin/Cummings.

Boshier, R. & Wilson, M.. (1998). *Panoptic Variations: Surveillance and Discipline in Web Courses*. Paper presented to the Adult Education Research Conference 1998, San Antonio, TX.

Bowers, C. A. (2000). *Let Them Eat Data: How Computers Affect Education, Cultural Diversity, and the Prospects of Ecological Sustainability.* Athens: University of Georgia Press.

Bowker, G. & Star, S. L. (1999). *Sorting Things Out: Classification and its Consequences.* Cambridge: MIT Press.

Brand, S. (1972, December). Spacewar: Fanatic Life and Symbolic Death Among the Computer Bums. *Rolling Stone*.

Brand, S. (1987). *The Media Lab: Inventing the Future at MIT.* New York: Viking.

Bricmont, J. & Sokal, A. (2001). Remarks on Methodological Relativism and "Antiscience." In J. A. Labinger & H. Collins (Eds.), *The One Culture: A Conversation about Science* (pp. 179-183). Chicago: University of Chicago Press.

Bromley, H. & Apple, M. (Eds.). (1998). *Education/Technology/Power: Educational Computing as a Social Practice.* Albany: State University of New York Press.

Brown, J. S. & Duguid, P. (1996). The Social Life of Documents. *First Monday, 1*(1).

Bruckman, A. S. (1994). *Programming for Fun: MUDs as a Context for Collaborative Learning*. Paper presented at the National Educational Computing Conference, Boston, MA.

Bruckman, A. S. (1997). *MOOSE Crossing*. Unpublished doctoral dissertation, Massachussetts Institute of Technology.

Bruner, J. S. (1966). *Toward a Theory of Instruction.* Cambridge: Harvard University Press.

Bruner, J. S. (1990). *Acts of Meaning.* Cambridge: Harvard University Press.

Bruner, J. S. (1991). The Narrative Construction of Reality. *Critical Inquiry, 18*(1), 1–21.

Bruner, J. (1996). *The Culture of Education*. Cambridge: Harvard University Press.

Bryson, M. & Castell, S. d. (1994). Telling Tales Out of School: Modernist, Critical, and "True Stories" about Educational Computing. *Journal of Educational Computing Research, 10*(3), 199–221.

Bryson, M. & de Castell, S. (1998). New Technologies and the Cultural Ecology of Primary Schooling: Imagining Teachers as Luddites In/Deed. *Educational Policy, 12*(5), 542–567.

Burk, J. (1998). The Play's the Thing: Theatricality and the MOO Environment. In C. Haynes & J. R. Holmevik (Eds.), *High Wired: On the Design, Use, and Theory of Educational MOOs.* Ann Arbor: University of Michigan Press.

Callon, M. (1981). Struggles and Negotiations to Define What is Problematic and What is Not: The Socio-logic of Translation. In K. D. Knorr, R. Krohn, & R. Whitley (Eds.), *The Social Process of Scientific Investigation* (pp. 197-219). Dordrecht: D. Reidel.

Callon, M. (1986). Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St. Brieuc Bay. In J. Law (Ed.), *Power, Action and Belief: A New Sociology of Knowlege?* (pp. 196-233). London: Routledge & Kegan Paul.

Callon, M. (1991). Techno-economic Networks and Irreversibility. In J. Law (Ed.), *A Sociology of Monsters: Essays on Power, Technology, and Domination.* London: Routledge.

Callon, M. & Latour, B. (1981). Unscrewing the Big Leviathan: How Actors Macro-structure Reality and How Sociologists Help Them to Do So. In K. Knorr-Cetina & A. V. Cicourel (Eds.), *Advances in Social Theory and Methodology: Toward an Integration of Micro- and Macro-sociologies.* Boston: Routledge & Kegan Paul.

Canadian Internet Project. (2005). *Canada Online.* Retrieved February 5, 2006, from http://www.cipic.ca

Carr, D. (1998). Narrative and the Real World: An Argument for Continuity. In B. Fay, P. Pomper, & R. T. Vann (Eds.), *History and Theory: Contemporary Readings* (pp. 137-152). Oxford: Blackwell Publishers.

de Castell, S., Bryson, M., & Jenson, J. (2002). Object Lessons: Towards an Educational Theory of Technology. *First Monday, 7*(1). Retrieved October 5, 2005, from http://www.firstmonday.org/issues/issue7_1/castell

Chakraborty, A., Graebner, R., & Stocky, T. (1999). *Logo: A Project History.* Unpublished paper for MIT's 6.933 course.

Chesley, H. et. al, . (1994). *End User Programming: Discussion of Fifteen Ideals*. RN-94-13. Apple Computer, Inc..

Cohen, R. (1986). History and Genre. *New Literary History 17*, 203–218.

Cole, M. (1996). *Cultural Psychology: A Once and Future Discipline.* Cambridge: Harvard University Press.

Collins, H, & Yearly, S. (1992). Epistemological Chicken. In A. Pickering (Ed.), *Science as Practice and Culture*. (pp. 301–326). Chicago: University of Chicago Press.

Cringley, R. X. *(1996). Triumph of the Nerds*, Part 3. Television Broadcast, PBS.

Cuban, L. (1986). *Teachers and Machines: The Classroom use of Technology Since 1920.* New York: Teachers College Press.

Cuban, L. (1993). Computers Meet Classroom: Classroom Wins. *Teachers College Record 95*(2), 185–210.

Cuban, L. (2001). *Oversold and Underused: Computers in the Classroom.* Cambridge: Harvard University Press.

Curtis, P. (1992). *Mudding: Social Phenomena in Text-Based Virtual Realities.* Paper presented at the Directions and Implications of Advanced Computing conference. Retrieved October 7, 1996, from ftp://parcftp.xerox.com/pub/MOO/papers/DIAC92.txt

Davy, J. (1985). Mindstorms in the Lamplight. In D. Sloan (Ed.), *The Computer in Education: A Critical Perspective.* New York: Teachers College Press.

Dewey, J. (1916). *Democracy and Education: An Introduction to the Philosophy of Education.* New York: The Macmillan Company.

diSessa, A. A. (2000). *Changing Minds: Computers, Learning, and Literacy.* Cambridge: MIT Press.

Dijkstra, E. W. (1989, December). On the Cruelty of Really Teaching Computer Science. *Communications of the ACM, 32*(12).

Doyle, M. (1995). An Informatics Curriculum. *InTegrate 19*. Retrieved October 5, 2005, from http://acitt.digitalbrain.com/acitt/web/resources/pubs/Integrate%2019/informatics%20curric.htm

Drucker, J. (1995). *The Alphabetic Labyrinth: The Letters in History and Imagination*. London: Thames & Hudson.

Ducasse, S. (2005). *Squeak: Learn Programming with Robots*. Berkeley: APress.

Dugan, B. (1994). *Simula and Smalltalk: A Social and Political History*. Retrieved Cotober 5, 2005, from http://www.cs.washington.edu/homes/dugan/history.html

Edwards, P. N. (1996). *The Closed World: Computers and the Politics of Discourse in Cold War America.* Cambridge: MIT Press.

Ehn, P. (1988). *Work-Oriented Design of Computer Artifacts*. Stockholm: Arbitslivscentrum.

Engelbart, D. & English, W. (2004). A Research Center for Augmenting Human Intellect. In N. Wardrip-Fruin & N. Montfort. (Eds.), *The New Media Reader* (pp. 93–105). Cambridge: MIT Press. (Original work published 1968)

Feenberg, A. (1999). *Questioning Technology.* New York: Routledge.

Feenberg, A. (2004). Democratic Rationalization: Technology, Power, and Freedom. In D. M. Kaplan (Ed.), *Readings in the Philosophy of Technology* (pp. 209-226). Lanham, MD: Rowman and Littlefield.

Fenton, J. & Beck, K. (1989). *Playground: An Object Oriented Simulation System with Agent Rules for Children of All Ages.* Paper presented at OOPSLA '89.

Fish, S. (1980). *Is there a Text in this Class?* Cambridge: Harvard University Press.

Franklin, U. M. (1999). *The Real World of Technology*. Revised Edition. Toronto: Anansi.

Friesen, N. (2001). What are Educational Objects? *Interactive Learning Environments, 9*(3), 219–230.

Fullan, M. & Hargreaves, A. (1996). *What's Worth Fighting for in Your School* (Second ed.). New York: Teachers College Press.

Gabriel, R. (1991). The Rise of 'Worse is Better'. In *Lisp: Good News, Bad News, How to Win Big*. Lucid, Inc. Retrieved October 5, 2005, from http://www.dreamsongs.com/WIB.html

Gadamer, H-G. (1999). *Truth and Method* (Second, Revised ed.; J. Weinsheimer & D. G. Marshall, Trans). New York: Continuum. (Originally published in English 1975; in German 1960)

Galas, C. (2001, October). School Squeaking. *SqueakNews, 1*(4).

Gardner, H. (1985). *The Mind's New Science: A History of the Cognitive Revolution*. New York: Basic Books.

Gardner, H. (1993). *Frames of Mind: The Theory of Multiple Intelligences*. New York: Basic Books.

Gardner, M. (1970, October). The Fantastic Combinations of John Conway's New Solitaire Game "Life." *Scientific American 223*, 120–123.

Garfinkel, H. (1967). *Studies in Ethnomethodology*. Englewood Cliffs: Prentice-Hall.

Geertz, C. (1973). *The Interpretation of Cultures.* New York: Basic Books.

Gillespie, T. (2002, Fall.). Hard Fun... Squeak! *Technos Quarterly, 11*(3).

Goldberg, A. (1979). Educational Uses of a Dynabook. *Computers & Education 3*, 247–266.

Goldberg, A. (1984). *Smalltalk-80: The Interactive Programming Environment*. Reading, MA: Addison-Wesley.

Goldberg, A. (Ed.). (1988). A History of Personal Workstations. Reading, MA: Addison-Wesley.

Goldberg, A. (1998). The Community of Smalltalk. In P. H. Salus (Ed.), *Handbook of Programming Languages, Volume 1: Object-Oriented Programming Languages* (pp. 51-94). New York: MacMillan Technical Publishing.

Goldberg, A. & Kay, A. C. (Eds.). (1976). *Smalltalk-72 Instruction Manual.* Xerox Palo Alto Research Center.

Goldberg, A. & Robson, D. (1983). *Smalltalk-80: The Language and its Implementation.* Reading, MA: Addison-Wesley.

Goldberg, A. & Ross, J. (1981, August). Is the Smalltalk-80 System for Children? *BYTE 6*(8) , 348–368.

Goldberg, A., Abell, S. T., & Leibs, D. (1997). The LearningWorks Development and Delivery Frameworks. *Communications of the ACM, 40*(10), 78–81.

Goldfarb, C. & Rubinsky, Y. (1991). *The SGML Handbook.* Oxford: Oxford University Press.

Goldman-Segall, R. (1995). Configurational Validity: A Proposal for Analyzing Ethnographic Multimedia Narratives. *Journal of Educational Multimedia and Hypermedia, 4*(2/3). pp. 163-182.

Goldman-Segall, R. (1998). *Points of Viewing Children's Thinking.* Hillsdale, NJ: Lawrence Erlbaum and Associates.

Goldman-Segall, R. & Maxwell, J. W. (2003). Computers, the Internet, and New Media for Learning. In W. M. Reynolds & G. E. Miller (Eds.), *Handbook of Psychology, vol. 7: Educational Psychology* (pp. 393-427). New York: John Wiley & Sons.

Graham, P. (1993). *On Lisp: Advanced Techniques for Common Lisp.* New York: Prentice Hall.

Graham, P. (2001). *The Roots of Lisp.* Retrieved October 5, 2005, from http://www.paulgraham.com/rootsoflisp.html

Graves, W. I. (1995). Ideologies of Computerization. In M. A. Sheilds (Ed.), *Work and Technology in Higher Education: The Social Construction of Academic Computing* (pp. 65-87). Hilldale, NJ: Lawrence Erlbaum Associates.

Graves, W. H. (1999). The Instructional Management Systems Cooperative: Converting Random Acts of Progress into Global Progress. *Educom Review, 34*(6).

Greenberg, D. (2001). The Trojan Horse of Education. *Technos Quarterly, 10*(1).

Grint, K. & Woolgar, S. (1997). *The Machine at Work: Technology, Work, and Organization.* Cambridge: Polity Press.

Grumet, M. (1988). *Bitter Milk: Women and Teaching.* Amherst: University of Massachusetts Press.

Groff, D. & Steele, K. (2004). *A Biased History of Interactive Design*. Smackerel. Retrieved January 5, 2006, from http://www.smackerel.net/smackerel_home.html

Guzdial, M. (1997). OOPSLA 97 Notes. Retrieved October 5, 2005, from http://www-static.cc.gatech.edu/fac/mark.guzdial/squeak/oopsla.html

Guzdial, M. (2000). *Squeak: Object Oriented Design with Multimedia Applications*. New York: Prentice Hall.

Guzdial, M. & Rose, K. M. (2001). *Squeak: Open Personal Computing and Multimedia*. New York: Prentice Hall.

Habermas, J. (1984). *The Theory of Communicative Action, Volume 1: Reason and the Rationalization of Society* (T. McCarthy, Trans.). Boston: Beacon Press.

Hadamard, J. (1954). *An Essay on the Psychology of Invention in the Mathematical Field.* New York: Dover.

Hamilton, A., Madison, J., & Jay, J. (1987). *The Federalist Papers* (I. Kramnick, Ed.). New York: Penguin.

Harasim, L. (1990). *Online Education: Perspectives on a New Environment*. New York: Praeger.

Haraway, D. J. (1991). *Simians, Cyborgs, and Women: The Reinvention of Nature.* New York: Routledge.

Haraway, D. J. (1997). *Modest_Witness@Second_Millennium.FemaleMan©_Meets_OncoMouse™.* New York: Routledge.

Harel, I. & Papert, S. (Eds.). (1991). *Constructionism.* Norwood, NJ: Ablex Publishing.

Harvey, B. (1991). *Symbolic Programming vs. the A.P. Curriculum.* Cambridge, MA: The Logo Foundation. Retrieved October 5, 2005, from http://el.media.mit.edu/Logo-foundation/pubs/papers/symbolic_vs_ap.html

Havelock, E. A. (1980). The Coming of Literate Communication to Western Culture. *Journal of Communication, 30*(1), 90–95.

Hayles, N. K. (2005). *My Mother Was a Computer: Digital Subjects and Literary Texts*. Chicago and London: University of Chicago Press.

Haynes, C. & Holmevik, J. R. (Eds.). (1998). *High Wired: On the Design, Use, and Theory of Educational MOOs*. Ann Arbor: University of Michigan Press.

Heidegger, M. (1993). The Question Concerning Technology. W. Lovitt (Trans.). In D. F. Krell (Ed.), *Basic Writings*. Revised and Expanded Edition. San Francisco: HarperCollins. (Originally published 1953)

Hemetsberger, A. & Reinhardt, C. (2004). *Sharing and Creating Knowledge in Open-Source Communities: The Case of KDE*. Paper presented at the Fifth European Conference on Organizational Knowledge, Learning, and Capabilities , Innsbruck 2004.

Hiltz, S. R. & Turoff, M.. (2000). *Breaking Down the Walls of the Classroom: A Tutorial on Distance Learning*. Paper presented at the AIWoRC '00 Conference & Expo on Virtual Organizations.

Hiltzik, M. A. (1999). *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age.* New York: HarperBusiness.

Hobart, M. E. & Schiffman, Z. S. (1998). *Information Ages: Literacy, Numeracy, and the Computer Revolution.* Baltimore: Johns Holpkins Press.

Horn, B. (n.d.). *Joining the Mac Group.* Retrieved February 5, 2006, from http://folklore.org/StoryView.py?project=Macintosh&story=Joining_the_Mac_Group.txt

Huhtamo, E. (1995). *From Kaleidoscomaniac to Cybernerd. Towards an Archeology of the Media*. Retrieved January 5, 2006, from http://www.debalie.nl/artikel.jsp?articleid=10104

IMS Project. (2002). *Instructional Management System Global Learning Consortium*. Project website. Retrieved October 7, 2004, from http://www.imsproject.org/.

Ingalls, D. H. H. (1978). *The Smalltalk-76 Programming System Design and Implementation.* Paper presented to the ACM Symposium on Principles of Programming Languages.

Ingalls, D. H. H. (1981, August). Design Principles Behind Smalltalk. *BYTE 6*(8), 286–298.

Ingalls, D. H. H. (1983). The Evolution of the Smalltalk Virtual Machine. In G. Krasner (Ed.), *Smalltalk-80: Bits of History, Words of Advice* (pp. 9-28). Reading, MA: Addison-Wesley.

Ingalls, D. H. H. (1996). The Birth of Squeak. Message posted to the Usenet group comp.lang.smalltalk. Archived at http://minnow.cc.gatech.edu/squeak/1985

Ingalls, D. H. H. (1997). Where Squeak Was Headed. Retrieved Oct 5, 2005, from http://www.squeak.org/about/headed-99-98.html

Ingalls, D. H. H. (2001, Sept). Squeak's 5th Birthday: An E-View with Dan Ingalls. *SqueakNews, 1*(3).

Ingalls, D. H. H., Kaehler, T., Maloney, J., Wallace, S., & Kay, A.. (1997). Back to the Future: The Story of Squeak, A Practical Smalltalk Written in Itself. Paper presented at OOPSLA '97. Retrieved October 5, 2005, from http://users.ipa.net/~dwighth/squeak/oopsla_squeak.html

Jauss, H. R. (1982). *Toward an Aesthetic of Reception* (T. Bahti, Trans). Minneapolis: University of Minnesota Press.

Kay, A. C. (1968). *The Reactive Engine.* Unpublished doctoral dissertation, University of Utah.

Kay, A. C. (1972). *A Personal Computer for Children of All Ages.* Paper presented at the ACM National Conference, Boston.

Kay, A. C. (1977, September). Microelectronics and the Personal Computer. *Scientific American 237, 230–244.*

Kay, A. C. (1978, March). *Ideas for Novice Programming in a Personal Computing System*. Paper presented to the Infotech State-of-the-Art Conference on User-friendly Systems, London.

Kay, A. C. (1979). Programming Your Own Computer. In *Science Year 1979* (pp. 183–195). World Book Encyclopedia.

Kay, A. C. (1984, September). Computer Software. *Scientific American, 251*(3), 53–59.

Kay, A. C. (1987). *Doing with Images Makes Symbols.* Video presentation. University Video Communications.

Kay, A. C. (1990). User Interface: A Personal View. In B. Laurel & S. J. Mountford (Eds.), *The Art of Human-Computer Interface Design* (pp. 191–207). Reading, MA: Addison-Wesley..

Kay, A. C. (1991, September). Computers, Networks and Education. *Scientific American 265*, 138–148.

Kay, A. C. (1994, September/October). Which Way to the Future? *Earthwatch: The Journal of Earthwatch Institute, 13*(5), 14–18.

Kay, A. C. (1995, October 12). *Powerful Ideas Need Love Too!* Written remarks to Joint Hearing on Educational Technology in the 21st Century, Science Committee and the Economic and Educational Opportunities Committee, US House of Representatives, Washington D.C.

Kay, A. C. (1996a). The Early History of Smalltalk. In T. J. Bergin & R. G. Gibson (Eds.), *History of Programming Languages II* (pp. 511-578). New York: ACM Press.

Kay, A. C. (1996b, Fall). Alan Kay Lecture. Multimedia Pioneers Series. San Francisco State University.

Kay, A. C. (1996c, July/August). Revealing the Elephant: The Use and Misuse of Computers in Education. *Educom Review, 31*(4).

Kay, A. C. (1997, July). Technology and Powerful Ideas: The real value of computers is in helping us understand the powerful ideas that force us to change our ways of thinking. *American School Board Journal, 97*, 16–19.

Kay, A. C. (1998a, December 11). Configuring a Dynabook. Message posted to Squeak-dev mailing list.

Kay, A. C. (1998b, October 10). Re: Prototypes vs. Classes. Message posted to Squeak-dev mailing list. Archived at http://lists.squeakfoundation.org/pipermail/squeak-dev/1998-October/017019.html

Kay, A. C. (1999, March/April). Software Design, the Future of Programming and the Art of Learning. *Educom Review, 34*(2).

Kay, A. C. (2000a). Dynabooks: Past, Present, and Future. *Library Quarterly, 70*(3), 385-395.

Kay, A. C. (2000b). Software: Art, Engineering, Mathematics, or Science?. In M. Guzdial (Ed.), *Squeak: Object-oriented Design with Multimedia Applications* (pp. xi-xiii). Upper Saddle River: Prentice Hall.

Kay, A. C. (2001, October 18). Lots of concurrency. Message posted to the squeak-dev mailing list. Archived at http://lists.squeakfoundation.org/pipermail/squeak-dev/2001-October/029823.html

Kay, A. C. (2002a, March). *The Computer Revolution Hasn't Happened Yet.* Lecture given to the Digital Cultures Project: Interfacing Knowledge: New Paradigms for Computing in the Humanities, Arts and Social Sciences. UC Santa Barbara.

Kay, A. C. (2002b). The Dynabook Revisited: A Conversation with Alan Kay. *The Book and the Computer*. Retrieved April 5, 2005, from http://www.honco.net/os/kay.html

Kay, A. C. (2003a). *Daddy Are We There Yet?*. Presentation to O'Reilly & Associates Emerging Technologies Conference. Retrieved October 5, 2005, from http://www.lisarein.com/videos/oreilly/etech2003/alankay/tour.html

Kay, A. C. (2003b, March). Etoys history. Message posted to squeak-dev mailing list.

Kay, A. C. (2004a). *The Power of the Context.* Remarks upon being awarded the Charles Stark Draper Prize of the National Academy of Engineering, Feb 24, 2004.

Kay, A. C. (2004b). *The Last Time You Thought Clearly Was*. Retrieved January 5, 2006, from http://squeakland.org/school/HTML/essays/habit_of_mind.htm

Kay, A. C. (2004c, October). *ACM Turing Lecture: Introductions To Computing Should Be Child's Play*. Lecture given at OOPSLA '04, Vancouver.

Kay, A. C. & Feldman, S. (2004, December/January). A Conversation with Alan Kay. *ACM Queue, 2*(9).

Kay, A. C. & Goldberg, A. (1976). *Personal Dynamic Media.* Xerox Corporation.

Kelleher, C. & Pausch, R. (2005, June). Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys 37*(2). 83–137.

Kellner, D. (2006). Technological Transformation, Multiple Literacies, and the Re-visioning of Education. In J. Weiss, J. Nolan, J. Hunsinger, & P. Trifonas (Eds.), *The International Handbook of Virtual Learning Environments*. pp. 241–268. Dordrecht: Springer.

Kimber, E. (1998). *Practical Hypermedia: An Introduction to HyTime* (Review ed.). Retrieved October 7, 1999, from http://www.drmacro.com/bookrev/practhyt/practicalhypermedia.html

Kincheloe, J. & Steinberg, S. R. (Eds.) (1992). *Thirteen Questions : Reframing Education's Conversation* New York : Peter Lang.

Kittler, F. (1995, October). There is No Software. *CTheory.net*. Retrieved January 5, 2006, from http://www.ctheory.net/articles.aspx?id=74

Klassen, P., Maxwell, J.W., & Norman, S. (1999). *Structured Information and Course Development: An SGML/XML Framework for Open Learning.* Paper presented at ED-MEDIA: World Conference on Educational Multimedia, Hypermedia and Telecommunications, Seattle, WA.

Koestler, A. (1964). *The Act of Creation*. New York: Macmillan.

Koschmann, T. (1996). Paradigm Shifts and Instructional Technology: An Introduction. In T. Koschmann (Ed.), *CSCL: Theory and Practice of an Emerging Paradigm.* Mahwah, NJ: Lawrence Erlbaum Associates.

Krasner, G. (1983). *Smalltalk-80: Bits of History, Words of Advice*. Reading, MA: Addison-Wesley.

Kuhn, T. S. (1996). *The Structure of Scientific Revolutions*. 3rd Edition. Chicago: University of Chicago Press.

Landow, G. P. (1992). *Hypertext: The Convergence of Contemporary Critical Theory and Technology*. Baltimore: Johns Hopkins University Press.

Lanier, J. (2006). The Gory Antigora: Illusions of Capitalism and Computers. *CATO Unbound*, January 2006. Retrieved August 5, 2006, from http://www.cato-unbound.org/2006/01/09/jaron-lanier/the-gory-antigora/

Laszlo, E. (1972). *The Systems View of the World.* New York: George Braziller.

Lather, P. (1991). Issues of Validity in Openly Ideological Research: Between a Rock and a Soft Place. *Interchange, 17*(4), 63–84.

Latour, B. (1987). *Science in Action.* Cambridge: Harvard University Press.

Latour, B. (1992). Where are the Missing Masses? The Sociology of a Few Mundane Artifacts. In W. E. Bijker & J. Law (Eds.), *Shaping Technology/Building Society: Studies in Sociotechnical Change.* Cambridge: MIT Press.

Latour, B. (1993). *We Have Never Been Modern* (C. Porter, Trans). Cambridge: Harvard University Press.

Latour, B. (1996). *Aramis, or the Love of Technology.* Cambridge: Harvard University Press.

Latour, B. (1999). *Pandora's Hope: Essays on the Reality of Science Studies.* Cambridge: Harvard University Press.

Latour, B. (2003). The Promises of Constructivism. In D. Ihde & E. Selinger (Eds.), *Chasing Technoscience: Matrix for Materiality.* Bloomington: Indiana University Press.

Latour, B. & Woolgar, S. (1986). *Laboratory Life: The Construction of Scientific Facts* Second Edition. Princeton: Princeton University Press. (Original edition 1979)

Laurel, B. & Mountford, S. J. (Eds.). (1990). *The Art of Human-Computer Interface Design*. Reading, MA: Addison-Wesley.

Laurel, B. (1993). *Computers as Theatre*. Reading, MA: Addison-Wesley.

Lave, J. (1988). *Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life.* Cambridge: Cambridge University Press.

Lave, J. & Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation.* Cambridge: Cambridge University Press.

Law, J. (1992). *Notes on the Theory of the Actor Network: Ordering, Strategy and Heterogeneity*. Centre for Science Studies, Lancaster University. Retrieved October 5, 2005, from http://www.comp.lancs.ac.uk/sociology/papers/Law-Notes-on-ANT.pdf

Lemke, J. (2001). *Towards a Theory of Traversals.* Retrieved January 30, 2006, from http://academic.brooklyn.cuny.edu/education/jlemke/papers/traversals/traversal-theory.htm

Lessig, L. (1999). *Code: And Other Laws of Cyberspace*. New York: Basic Books.

Lessig, L. (2002a). *Free Culture.* Retrieved February 5, 2006, from http://randomfoo.net/oscon/2002/lessig/

Lessig, L. (2002b). *The Future of Ideas: The Fate of the Commons in a Connected World.* New York: Vintage Books.

Licklider, J. C. R. (1960, March). Man-Computer Symbiosis. *IRE Transactions, HFE-1*, 4-11.

Lombardi, M. M. (2004, April). Croquet Anyone? Designing a More Responsive Online Learning Environment. *Teaching With Technology Today, 10*(5).

Lowry, M. (1979). *The World of Aldus Manutius: Business and Scholarship in Renaissance Venice.* Ithaca: Cornell University Press.

Lyman, P. (1995). Is Using a Computer Like Driving a Car, Reading a Book, or Solving a Problem? The Computer as Machine, Text, and Culture. In M. A. Shields (Ed.), *Work and Technology in Higher Education: The Social Construction of Academic Computing* (pp. 19-36). Hillsdale, NJ: Lawrence Erlbaum Associates.

MacIntyre, A. (1984). *After Virtue* (Second ed.). Notre Dame: University of Notre Dame Press.

Makreel, R. A. (2004). An Ethically Responsive Hermeneutics of History. In D. Carr, T. R. Flynn, & R. A. Makkreel (Eds.), *The Ethics of History.* Evanston: Northwestern University Press.

Maloney, J. (2001, August). Interview with John Maloney Part II: The Apple Days. *SqueakNews, 1*(2).

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). *Scratch: A Sneak Preview*. Paper presented at the Second International Conference on Creating, Connecting, and Collaborating through Computing. Kyoto, Japan. Retrieved February 5, 2006, from http://llk.media.mit.edu/projects/scratch/ScratchSneakPreview.pdf

Manovich, L. (2003). New Media from Borges to HTML. In N. Wardrip-Fruin & N. Montfort (Eds.), *The New Media Reader* (pp. 13–25). Cambridge: MIT Press.

Margolis, J. & Fisher, A. (2002). *Unlocking the Clubhouse: Women in Computing*. Cambridge: MIT Press.

Marion, A. (1993). *Playground Paper: Reviewing an experiment in designing a computer programming environment for children within an elementary school.* Unpublished report: Apple Computer's Advanced Technology Group.

Markoff, J. (2005). *What the Dormouse Said: How the 60s Counterculture Shaped the Personal Computer Industry.* New York: Viking.

Maxwell, J.W. (1996). *House of Words: Designing Text and Community in Multi-User Object-Oriented Environments.* Unpublished masters thesis, Simon Fraser University.

McCarthy, J. (1960, April). Recursive Functions of Symbolic Expressions and Their Computation by Machine. *Communications of the ACM, 3*(4).

McCarthy, J. (1981). The History of LISP. In R. L. Wexelblat (Ed.), *History of Programming Languages.* New York: Academic Press.

McCullough, M. (1998). *Abstracting Craft: The Practiced Digital Hand.* Cambridge: MIT Press.

McLaren, P. (1993). *Schooling as a Ritual Performance : Towards a Political Economy of Educational Symbols and Gestures* (Second ed.). New York: Routledge.

McLuhan, M. (1962). *The Gutenberg Galaxy: The making of Typographic Man.* New York: Routledge & Kegan Paul.

McLuhan, M. (1964). *Understanding Media: The Extensions of Man.* New York: McGraw-Hill.

Meeker, M. (2005). *Internet Trends*. Retrieved February 5, 2006, from http://www.morganstanley.com/institutional/techresearch/pdfs/GSB112005.pdf

Menzies, H. (1989). *Fast Forward and Out of Control: How Technology is Changing Your Life*. Toronto: Macmillan.

Menzies, H. (1999). Digital Networks: The Medium of Globalization, and the Message. *Canadian Journal of Communication 24(4)*

Miller, C. R. (1994a). Genre as Social Action. In A. Freedman & P. Medway (Eds.), *Genre and the New Rhetoric* (pp. 23–42). London: Taylor and Francis. (Originally published 1984)

Miller, C. R. (1994b). Rhetorical Community: The Cultural Basis of Genre. In A. Freedman & P. Medway (Eds.), *Genre and the New Rhetoric* (pp. 67-78). London: Taylor and Francis.

Miller, J. A. (2004). *Promoting Computer Literacy Through Programming Python.* Unpublished doctoral dissertation, University of Michigan.

Montessori, M. (1972). *The Discovery of the Child* (M. J. Costelloe, Trans). New York: Ballantine Books.

Moglen, E. (2000, Winter). The Encryption Wars: An Interview with Jay Worthington. *Cabinet 1* . Retrieved June 6, 2005, from http://www.cabinetmagazine.org/issues/1/i_moglen_1.hphp

Moglen, E. (2003, June). *Freeing the Mind: Free Software and the Death of Proprietary Culture*. Paper presented at the University of Maine Law School's Fourth Annual Technology and Law Conference.

Müller-Prove, M. (2002). *Vision and Reality of Hypertext and Graphical User Interfaces.* Unpublished masters thesis, Universität Hamburg.

National Research Council. (1999). *Funding a Revolution: Government Support for Computing Research.* Washington, DC: National Academy Press.

New London Group. (1996). A Pedagogy of Multiliteracies: Designing Social Futures. *Harvard Education Review 66*(1), 60–92.

Noble, D. (1999). *Digital Diploma Mills Part IV: Rehearsal for the revolution.* Retreived Oct 7, 2003, from http://communication.ucsd.edu/dl/ddm4.html

Norman, A. P. (1998). Telling It Like It Was: Historical Narratives On Their Own Terms. In B. Fay, P. Pomper, & R. T. Vann (Eds.), *History and Theory: Contemporary Readings* (pp. 119-135). Oxford: Blackwell Publishers.

Noss, R. & Hoyles, C. (1996). *Windows on Mathematical Meanings.* Kluwer.

Ong, W. J. (1995). *Orality and Literacy: The Technologizing of the Word*. London and New York: Routledge.

Oppenheimer, T. (1997, July 1). The Computer Delusion. *The Atlantic Monthly, 280,* 45-62.

O'Reilly, T. (1999). Hardware, Software, and Infoware. In C. DiBona, S. Ockman, & M. Stone (Eds.), *Open Sources: Voices from the Open Source Revolution*, First ed. Sebastopol: O'Reilly & Associates.

Pacey, A. (1983). *The Culture of Technology.* Oxford: Basil Blackwell.

Papert, S. (1980a). *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books.

Papert, S. (1980b). Teaching Children Thinking. In R. Taylor (Ed.), *The Computer in the School: Tutor, Tool, Tutee* (pp. 160-176). New York: Teachers College Press. (Originally published 1972)

Papert, S. (1987). Information Technology and Education: Computer Criticism vs. Technocentric Thinking. *Educational Researcher, 16*(1), 22–30.

Papert, S. (1991). Situating Constructionism. In I. Harel & S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex Publishing.

Papert, S. (1992). *The Children's Machine.* New York: BasicBooks.

Papert, S. (1999, March 29). Papert on Piaget. *Time*. p. 105. Retrieved Jan 5, 2006, from http://www.papert.org/articles/Papertonpiaget.html

Pea, R. & Sheingold, K. (Eds.). (1987). *Mirrors of Minds.* Norwood NJ: Ablex.

Phillips, M. S. (2000). *Society and Sentiment: Genres of Historical Writing in Britain, 1740-1820.* Princeton: Princeton University Press.

Postman, N. (1970). The Reformed English Curriculum. In A. Eurich (Ed.), *High School 1980: The Shape of the Future in American Secondary Education.* New York: Pitman.

Postman, N. (1986). *Amusing Ourselves to Death: Public Discourse in the Age of Show Business*. New York: Penguin Books.

Prigogine, I. & Stengers, I. (1985). *Order out of Chaos: Man's New Dialogue with Nature.* London: Flamingo.

Raymond, E. S. (1999). *The Cathedral and the Bazaar*. version 1.46. Retrieved February 7, 2004, from http://tuxedo.org/~esr/writings/cathedral-bazaar/

Raymond, E. S. (2003). *The Art of Unix Programming*. New York: Addison-Wesley.

Rheingold. H. (1985). *Tools for Thought: The People and Ideas Behind the Next Computer Revolution*. Engleood Cliffs, NJ: Prentice-Hall

Ricoeur, P. (1991a). The Model of the Text: Meaningful Action Considered as Text. In J. B. Thompson & K. Blamey (Trans.), *From Text to Action: Essays in Hermeneutics, II.* Evanston, Ill: Northwestern University Press. (Originally published 1971)

Ricoeur, P. (1991b). The Hermeneutical Function of Distanciation. In J. B. Thompson & K. Blamey (Trans.), *From Text to Action: Essays in Hermeneutics, II*. Evanston, Ill: Northwestern University Press.

Rose, K. (2001, October). An In-Depth Interview with Kim Rose. *SqueakNews, 1*(4).

Rose, K. (2001, November). An In-Depth Interview with Kim Rose Pt 2. *SqueakNews, 1*(5).

Rose, E. (2003). *User Error: Resisting Computer Culture.* Toronto: Between the Lines .

SRI International. (1995). *The Los Angeles Open Charter School*. U.S. Department of Education. Retrieved March 8, 2005, from http://www.ed.gov/pubs/EdReformStudies/EdTech/opencharter.html

Saltzer, J. H., Reed, D. P., & Clark, D. D. (1984). End-to-end Arguments in System Design. *ACM Transactions on Computer Systems, 2*(4), 277-288.

Salus, P. H. (1994). *A Quarter Century of Unix*. Reading, MA: Addison-Wesley.

Schoch, J. F. (1979). An Overview of the Programming Language Smalltalk-72. A*CM SIGPLAN Notices, 14*(9).

Scollon, R. (2001). *Mediated Discourse: The Nexus of Practice.* London: Routledge.

Shields, M. A. (Ed.). (1995). *Work and Technology in Higher Education: The Social Construction of Academic Computing*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Sloan, D. (Ed.). (1985). *The Computer in Education: A Critical Perspective.* New York: Teachers College Press.

Smith, D. C. (1993). Pygmalion: An Executable Electronic Blackboard. In A. Cypher (Ed.), *Watch What I Do: Programming by Demonstration*. Cambridge: MIT Press.

Smith, D. K. & Alexander, R. C. (1988). *Fumbling the Future: How Xerox Invented, then Ignored, the First Personal Computer.* New York: Morrow.

Solomon, C. (1986). *Computer Environments for Children: A Reflection on Theories of Learning and Education.* Cambridge: MIT Press.

Stallman, R. (1985). The GNU Manifesto. *Dr. Dobbs Journal, 10*(3), 30–35.

Stallman, R. M. (1998). The GNU Operating System and the Free Software Movement. In C. DiBona, S. Ockham, & M. Stone (Eds.), *Open Sources: Voices from the Open Source Revolution*. Sebastopol: O'Reilly and Associates.

Stallman, R. M. (2001). *Copyright and Globalization in the Age of Computer Networks.* MIT Communications Forum. Retrieved February 5, 2006, from http://web.mit.edu/m-i-t/forums/copyright/transcript.html

Stallman, R. M. (2003). *Software Patents: Obstacles to Software Development*. Talk presented at the University of Cambridge Computer Laboratory. Retrieved October 5, 2005, from http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html

Star, S. L. (1999). The Ethnography of Infrastructure. *American Behavioral Scientist, 43*(3), 377–391.

Statistics Canada. (2004). *Connectivity and Learning in Canada's Schools*. Summarized report. Retrieved February 5, 2006, from http://www.pwgsc.gc.ca/onlineconsultation/text/statistics-e.html#Schools

Steele, G. & Gabriel, R. (1993, March). The Evolution of Lisp. *ACM SIGPLAN Notices 28*(3), 231–270.

Stephenson, N. (1999). *In the Beginning was the Command Line*. New York: Avon.

Stone, A. R. (1995). *The War of Desire and Technology at the End of the Mechanical Age*. Cambridge: MIT Press.

Stoyan, H. (1984). *Early LISP History.* Paper presented at the ACM Symposium on LISP and Functional Programming.

Stroustrup, B. (1998). A History of C++. In P. H. Salus (Ed.), *Handbook of Programming Languages, Volume I: Object-Oriented Programming Languages* (pp. 197–303). New York: MacMillan Technical Publishing.

Sutherland, I. (1963). *Sketchpad: A Man-machine Graphical Communication System*. Unpublished doctoral dissertation, Massachussetts Institute of Technology.

Swales, J. M. (1990). *Genre Analysis: English in Academic and Research Settings.* Cambridge: Cambridge University Press.

Tapscott, D. (2000). The Digital Divide. In *The Jossey-Bass Reader onTechnology and Learning*. San Francisco: Jossey-Bass.

Tate, B. (2005). *Beyond Java*. Sepastopol: O'Reilly Associates.

Thomas, D. (n.d.). Travels with Smalltalk. Retrieved Oct 5, 2005, from http://www.mojowire.com/TravelsWithSmalltalk

Torres, C. A. (Ed.) (1998). *Education, Power, and Personal Biography : Dialogues with Critical Educators*. New York: Routledge.

Travers, M. (1988). *Agar: An Animal Construction Kit*. Unpublished M.S. thesis, Massachussetts Institute of Technology.

Traweek, S. (1988). *Beamtimes and Lifetimes: The World of High Energy Physics.* Cambridge: Harvard University Press.

Tuomi, I. (2000). Internet, Innovation, and Open Source: Actors in the Network. *First Monday, 6*(1).

Turkle, S. (1984). *The Second Self: Computers and the Human Spirit.* New York: Simon and Schuster.

Turkle, S. (1995). Paradoxical Reactions and Powerful Ideas: Educational Computing in a Department of Physics. In M. A. Sheilds (Ed.), *Work and Technology in HIgher Education: The Social Construction of Academic Computing.* (pp. 37–64). Hillsdale, NJ: Lawrence Erlbaum Associates.

Turkle, S. & Papert, S. (1991). Epistemological Pluralism and the Revaluation of the Concrete. In I. Harel & S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex Publishing.

Tyler, S. A. (1986). Post-Modern Ethnography: From Document of the Occult to Occult Document. In J. Clifford & G. E. Marcus (Eds.), *Writing Culture*. Berkeley and Los Angeles: University of California Press.

Unsworth, J. (1995). *Living Inside the (Operating) System: Community in Virtual Reality* (Draft). Retrieved February 7, 2006, from http://www3.iath.virginia.edu/pmc/Virtual.Community.html

Vygotsky , L. S. (1978). Mind in Society: The Develpoment of Higher Psychological Processes. Cambridge: Harvard University Press.

Waldrop, M. M. (2001). *The Dream Machine : J. C. R. Licklider and the Revolution that Made Computing Personal.* New York: Viking.

Wardrip-Fruin, N. & Montfort, N. (Eds.). (2004). *The New Media Reader.* Cambridge: MIT Press.

Wall, L. (1999, August). *Perl, the First Postmodern Programming Language.* Paper presented to the Linux World Conference, San Jose CA.

Weizenbaum, J. (1976). *Computer Power and Human Reason: From Judgment to Calculation.* New York: W. H. Freeman and Company.

White, H. (1973). *Metahistory: The Historical Imagination in Nineteenth-Century Europe.* Baltimore and London: The Johns Hopkins University Press.

White, F. (1999). Digital Diploma Mills: A Dissenting Voice. *First Monday, 4*(7).

Whitehead, A. N. (1960). *Process and Reality: An Essay in Cosmology.* New Yrok: Harper.

WikiWikiWeb. (n.d.) Cunningham & Cunningham. Available at http://c2.com/cgi/wiki

Willinsky, J. (1998). *Learning to Divide the World: Education at Empire's End.* Minneapolis: University of Minnesota Press.

Winograd, T. & Flores, F. (1986). *Understanding Computer and Cognition: A New Foundation for Design.* Norwood, NJ: Ablex .

Wolfram, S. (2002). *A New Kind of Science*. Champaign, IL: Wolfram Media.

Woolley, D. R. (1994). PLATO: The Emergence of Online Community. *Computer-Mediated Communication Magazine, 1*(3). Retrieved March 8, 2005, from http://www.december.com/cmc/mag/1994/jul/

Yaeger, L. (1989). *Vivarium History*. Retrieved Oct 5, 2005, from http://homepage.mac.com/larryy/larryy/VivHist.html