Critical Programming: Toward a Philosophy of Computing

John Bork
University of Central Florida

CRITICAL PROGRAMMING
TOWARD A PHILOSOPHY OF COMPUTING

by

JOHN ROBERT BORK
B.A University of Houston, 1993
B.S. Bowling Green State University, 1999
M.I.T. Bowling Green State University, 2005

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Texts and Technology
in the College of Arts and Humanities
at the University of Central Florida
Orlando, Florida

Fall Term
2015

Major Professor: Bruce Janz

ABSTRACT

Beliefs about the relationship between human beings and computing machines and their destinies have alternated from heroic counterparts to conspirators of automated genocide, from apocalyptic extinction events to evolutionary cyborg convergences. Many fear that people are losing key intellectual and social abilities as tasks are offloaded to the 'everyware' of the built environment, which is developing a mind of its own. If digital technologies have contributed to forming a dumbest generation and ushering in a robotic moment, we all have a stake in addressing this collective intelligence problem. While digital humanities continue to flourish and introduce new uses for computer technologies, the basic modes of philosophical inquiry remain in the grip of print media, and default philosophies of computing prevail, or experimental ones propagate false hopes. I cast this as-is situation as the post-postmodern network dividual cyborg, recognizing that the rational enlightenment of modernism and regressive subjectivity of postmodernism now operate in an empire of extended mind cybernetics combined with techno-capitalist networks forming societies of control.

Recent critical theorists identify a justificatory scheme foregrounding participation in projects, valorizing social network linkages over heroic individualism, and commending flexibility and adaptability through life long learning over stable career paths. It seems to reify one possible, contingent configuration of global capitalism as if it was the reflection of a deterministic evolution of commingled technogenesis and synaptogenesis. To counter this trend I offer a theoretical framework to focus on the phenomenology of software and code, joining social critiques with textuality and media studies, the former proposing that theory be done through practice, and the latter seeking to understand their schematism of perceptibility by taking into account engineering techniques like time axis manipulation. The social construction of technology makes additional theoretical contributions dispelling closed world, deterministic historical narratives and requiring voices be given to the engineers and technologists that best know their subject area. This theoretical slate has been recently deployed to produce rich histories of computing, networking, and software, inform the nascent disciplines of software studies and code studies, as well as guide ethnographers of software development communities.

I call my syncretism of these approaches the procedural rhetoric of diachrony in synchrony, recognizing that multiple explanatory layers operating in their individual temporal and physical orders of magnitude simultaneously undergird post-postmodern network phenomena. Its touchstone is that the human-machine situation is best contemplated by doing, which as a methodology for digital humanities research I call critical programming. Philosophers of computing explore working code places by designing, coding, and executing complex software projects as an integral part of their intellectual activity, reflecting on how developing theoretical understanding necessitates iterative development of code as it does other texts, and how resolving coding dilemmas may clarify or modify provisional theories as our minds struggle to intuit the alien temporalities of machine processes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

From Automated Genocide to the Dumbest Generation

Histories of computing technologies often feature images of early electronic machinery developed in America and Britain during World War II, and connect their emergence to heroic narratives of triumphant, democratic civilization over genocidal, totalitarian regimes.[1] By a seamless progression of innovation, commercialization, and dissemination, these forerunners laid the ground for subsequent eras of mainframe, mini, personal, and networked computers, on to state of the art mobile, multimedia devices. However, in this dissertation that suggests, instead of another history, an approach *toward a philosophy of computing*, my inaugural image shall be a 1933 advertisement for Hollerith punch cards with which Edwin Black begins his provocatively titled *IBM and the Holocaust: The Strategic Alliance Between Nazi Germany and America's Most Powerful Corporation*. Rays emanating from an all-seeing eye illuminate an enormous punch card, factory, and smokestack, with the text *Ubersicht mit hollerith Lochkarten*. Black implicates IBM machinery, its employees, and its partners in America and Europe, with their bureaucratic counterparts in the murderous Nazi regime, embodying the evil latent in apparently benign technological devices.

IBM published but quickly withdrew a promotional book on the history of computing. *The History of Computing in Europe* purportedly detailed the exploits of famous employees on

1 For example, Paul E. Ceruzzi *A History of Modern Computing* and Michael R. Williams *History of Computing Technology*. Other books like Martin Campbell-Kelly *From Airline Reservations to Sonic The Hedgehog: A History of the Software Industry* use poster and magazine advertisements at the beginning of each chapter.

both the American and Nazi sides who were computer wizards of their day; a very rare book indeed, which Black claims is so rare as to not be found in any public library, or even Internet archives (425). Thus both a hidden history of the modern computing era and ripe store of unexamined philosophical perspectives awaits scholarly attention. My research devolves to the readily available material by early workers in the field, from Bush, Burks, Goldstein, von Neumann, Licklider, Kemeny, and so on, through Stallman, Knuth, Stroustrup, Gates, Torvalds, Jobs, and the like, yet this suppressed text symbolizes a holy grail of sorts.[2] Though perhaps now information technology gets imbricated in mass atrocities by dropping smart bombs from drones, rather than extracting innocents from the populace via census forms, herding them into ghettos, and operating death camps, it remains important to think about how information is gathered and processed, and how the fruits of human invention are used. Are the perpetrators as steeped in evil as the vilest hacker, are they morally ambivalent managers like Adolf Eichmann, or rank and file office workers blind to the purposes of their efforts – or perhaps the lists are already being made by machines on their own, leading to future genocides portrayed in science fiction apocalypses like *The Terminator*? That is one extreme perspective epitomized in science fiction, excellently argued by N. Katherine Hayles and a host of digital humanities theorists. However, I believe we

2 It is significant that that dreadful and mysterious IBM publication remains hidden, and that putative philosophers of computing should engage in scholarly quests to study it, in alignment with the nascent disciplines of software studies and critical code studies, which pursue philosophies implicitly and explicitly baked into program code statements and comments, software development practices, and so on, for which as a holy grail we posit the missing IBM text, and for accessible, everyday examples myriad projects of the Internet era found in source code revisions of content published under free, open source licenses. They yield a place to work, to discover and do philosophy, which I examine at length in chapters three and four.

are on a trajectory not aimed toward automated genocide, but rather unintentional stupefaction, reduction of potential in comparison to the increasing competence of machine cognition, despite enormous hopes and efforts by many well intentioned policy makers, engineers, and educators.

John Kemeny, inventor of the programming language BASIC in the 1960s, shared the optimism of J.C.R. Licklider and Douglas Engelbart that a great future of continuous improvement was in store for both humans and machines. Gerald Weinberg, who researched this period as a social psychologist, nonetheless concluded the default trajectory of the specific milieu in which our computer revolutions have occurred is tyranny over liberty, in spite of the good intentions of those we salute as the architects of the information age (279). In the final pages of the epilogue to his famous *Psychology of Computer Programming*, he warns of the danger of banality of evil through unwitting use of programming talent, though made in ironic, innocent ignorance of the real involvement of IBM in the holocaust with which my tale opens. He writes, “because computers are such fascinating beasts, because programming is such a game, such a joy, we who program computers are in danger of becoming the unwitting pawns of those who would use our toys for not-so-playful ends. Can there be any doubt that if Hitler had computers at his command, one of the first applications would have been keeping closer track of Jews and Gypsies so that all who should have gone to the ovens did go to the ovens?” (278).

Jaron Lanier argues such dangers lurk in so-called siren servers, and the case has been explicitly made in the recent video documentary *Terms and Conditions May Apply*. Is this a reflection of the need for a renewed critique and distancing oneself from technology? Toward a philosophy of computing, the overall humanities research question briefly cast, *What has been lost by humans while machines continue to evolve, and how can this trajectory be modified?*, shall be approached conceptually by considering the as-is situation of *post-postmodern network dividual cyborgs*. These are beings embodying the present condition in the United States, all of us – humans and our electronic devices – enacting network consumption, tightly coupled to the built environment, including increasingly intelligent machines, arrived at through decades of immersion in and constitution as expanding computer technologies in the overall context of late capitalist Internet age America, we who have, partly as a result, passed through a dumbest generation. In emphasizing extreme cases of hard-core programmers, Joseph Weizenbaum draws our attention away from the mundane, long term effects of using particular technologies, just as writers who analyze geek cultures shift focus from what has happened among a small percentage of professionals to everyday America, a condition of general stupefaction that Nietzsche referred to as the last man, that Horkheimer and Adorno decried as regressive bourgeois thinking (47).

To Langon Winner, the computer revolution has been influenced by an *absent mind* rather than new wonders emerging from artificial intelligence research, echoing the concern of philosophers and social scientists that democracy is not merely a matter of distributing information. The postmodern philosopher Frederic Jameson touches upon similar themes when he characterizes our comportment toward technology as a Promethean inferiority complex. Yet another social and technology critic, Neil Postman, proposed the notion of *technopoly* in 1993 for the unnamed, multi-spectrum force that alters the structure of human interests down to the level of our symbols, affecting the interactions of communities and whole populations (20).

It is from this vantage point that contemporary digital media theorist David Rushkoff argues that social hopes for the Internet seem to be failing, draining values and denying deep thinking rather than fostering highly articulated connections and new forms of creativity (16). To compound the damage on society as a whole, Lanier explains how the Internet has spanned an ecosystem of what he calls *siren servers,* which allow a select few to monetize the network usage of the masses. It is from such critical perspectives that I argue the decline in human intelligence, industriousness, and creativity – whose empirical validity is a research question but will be taken for granted – will not consummate in a regression to prehuman forms, a digital dark age, or machine apocalypse, but rather *leave behind traces suggesting that more advantageous synergies with machine intelligence could have been achieved*.[3]

My thesis is that the problem is complicated by humans getting dumber for want of spending time programming, or working code, replacing it with ordinary computer application use and other forms of so called direct manipulation. Thus the sinister Dehomag poster foreboding automated genocide gives way to imagery from the 2008 Disney movie *WALL-E,* of the evolutionary effects of generations lived in the machine-controlled spaceship environment of the *Axiom*. It depicts obese, shallowly content, physically and mentally unchallenged human consumers, whose needs are met and whose desires are fulfilled – precisely because they are also supplied and conditioned – by the surrounding intelligence of the built environment, over the watchful gaze of AUTO, their lives unfolding on screens aboard a gigantic cruise ship.[4] My basic

3 Optimistic counterexamples abound; however, as Bauerlein calls out champions of digital media, charter schools, and high-stakes testing for foregrounding the exceptional minority while ignoring the dire condition of the majority of American adolescents, technology critics like Winner and Feenberg downplay the importance of high profile success stories when measured against the dwindling position of the majority under hegemonic technopolies.

4 Philip Roth coined the term “the dumbest generation” in his 2000 novel *The Human Stain,* which Mark Bauerlein adopts for the title of his 2009 book *The Dumbest Generation: How the Digital Age Stupefies Young*

premise is that the dumbest generation has infected human being to steer it toward *WALL-E* torpor rather than apocalyptic science fiction narratives of automated destruction.[5]

Joseph Weizenbaum called them out: “Wherever computer centers have become established, that is to say, in countless places in the United States, as well as in virtually all other industrial regions of the world, bright young men of disheveled appearance, often with sunken glowing eyes, can be seen sitting at computer consoles, their arms tensed and waiting to fire their fingers, already poised to strike, at the buttons and keys on which their attention seems to be as riveted as a gambler's on the rolling dice. . . . Their rumpled clothes, their unwashed and unshaven faces, and their uncombed hair all testify that they are oblivious to their bodies and to the world in which they move. They exist, at least when so engaged, only through and for the computers. These are computer bums, compulsive programmers” (116). The compulsive programmer computer bums decried by Weizenbaum have been replaced and quantitatively outnumbered by hordes of compulsive gamers, social networkers, and others enveloped in consumer practices, their time wasted in front of the screen.

Americans and Jeopardizes Our Future, which he applies to the majority segment of young Americans entering adulthood ignorant and little concerned with liberal arts learning and civic awareness. My project therefore aligns with critiques of late capitalist societies from Allan Bloom's *Closing of the American Mind*, Luc Boltanski and Eve Chiapello's *The New Spirit of Capitalism*, through Catherine Malabou's *What Should We Do With Our Brain?* and Jaron Lanier's *Who Owns the Future?*, while adding consideration of the significant effects of dynamic media shot through with machine intelligence and pervasive automation by coded objects on bodies themselves and their extended but still closed minds, which I call the post-postmodern network dividual cyborg.

5 Examples abound from *2001: A Space Odyssey* and *Colossus: The Forbin Project* in the late 1960s and early 1970s, the *Terminator* trilogy in the mid 1980s through 1990s, the *Matrix* trilogy of 1999 through 2003, to the 2004 TV series *Battlestar Galactica* and its 2010 spin-off *Caprica*.

It used to be time wasted in front of papers and books. The research Mark Bauerlein cites in *The Dumbest Generation: How the Digital Age Stupefies Young Americans and Jeopardizes Our Future* appears to substantiate claims that most children in America spend more time with media than homework (4). Print literacy, the cornerstone of civilization, is being replaced with a dissimilar building block; imagination-inspiring books are supplanted by on-screen virtual realities. The threshold into adulthood has changed because the rituals that used to introduce it are shunted by the digital realm. This downward heading of the American mind towards *WALL-E* characters shall be the net effect of social pressures and leisure preferences, exacerbated by digital media. Whereas Bauerlein ultimately blames custodians of culture at all levels, from policy makers to educators, treating the influx of screen technologies as an aggravating circumstance rather than root cause, I want to explore further how we got ourselves into this collective intelligence problem, that more advantageous synergies with machine intelligence could have been achieved, and that we humans are unfortunately getting dumber while machines continue to get smarter. From there I will propose the discipline of critical programming as a means to alter that course.

A Collective Intelligence Problem

Henry Jenkins uses the term *collective intelligence* to name the contemporary collective process involving humans collaborating along with inhuman information technologies, especially Internet resources, together consuming and creating knowledge (4). The concept, on the one hand, seems essential for any knowledge to exist at all, implicating signs, symbols, and artifacts with biological entities, making all intelligence collective. That the nonhuman, technological component plays an active, participatory role, on the other hand, seems to be an emergent phenomenon. N. Katherine Hayles refers to the nondeterministic, evolutionary development of technology systems as *technogenesis*, and is adamant that it is deeply intertwined with concurrent *synaptogenesis*, the lifetime, peri-generational – rather than long term, epochal – changes in the human brains that use them (*How We Think* 11). John Kemeny, who invented the BASIC programming language with Thomas Kurtz, makes the implicit argument that through learning to program computers to perform the formerly mundane, repetitive knowledge work of prior generations, masses of humans who knowledgeably use information technologies will prosper in a new golden age. Such is his utopic vision of mutually augmenting synaptogenesis and technogenesis.

Such one-sided predictions of overall social benefits by technology evangelists recall the critique of writing in Plato. Thus Winner's critique of the conviction that widespread adoption of computers and communications systems will automatically produce a better world for human living, *mythinformation*, expresses the contemporary ideology that all aspects of life will likewise benefit from speedy digitized information processing, compounded by political assumptions of computer romantics mistaking the supply of information with the ability to leverage it. This is simply a false assumption that ordinary citizens equipped with microcomputers will be able to counter the influence of massive, computer-based organizations.[6] Postman also concludes that computers, like television, afford little to the masses, make no substantive, positive transformation of their condition, but instead primarily intrude on their

6 Winner makes an amusing analogy: “using a personal computer makes one no more powerful vis-a-vis, say, the National Security Agency than flying a hang glider establishes a person as a match for the U.S. Air Force" (“Mythinformation” 595).

lives, making the majority losers, and only a few winners. And as he says, “almost nothing that they need happens to the losers. Which is why they are losers” (10).

Societies of Control

At the heart of this collective intelligence problem are the cumulative effects of disciplinarity, interpellation, and production – i.e., Foucauldian biopower. To Deleuze, computers are emblematic of *societies of control*, which operate by continuous and short-term manipulation, in contrast to discontinuous, long duration actions of the disciplinary societies that Foucault studied. Control resembles a law of nature it is so imbricated in biopower and technological systems (Galloway 147). Rob Kitchin and Martin Dodge emphasize the positive, productive role of digital technologies, for “they make societies safer, healthier, and richer overall even as they do the work to regulate societies. . . . In Althusser's (1971) terms, software-driven technologies induce a process of interpellation, wherein people willingly and voluntarily subscribe to and desire their logic, trading potential disciplinary effects against benefits gained” (11). Yet Lawrence Lessig contends that legal rights to control cultural development are now more concentrated than ever because they are embedded in code controlled by large corporations (*Free Culture* 170). By implementing copyright law in the very mechanisms by which media is dispensed to consumers, code itself has become a pervasive form of control supplementing, if not overdetermining, its juridical and customary forms.

The ultimate consequence of societies of control is that, considered from the machine side of reality, human souls are inextricably encased in network phenomena with a life of their own. Media theorist and philosopher of computing Friedrich Kittler argues that “understanding media despite McLuhan's title remains an impossibility precisely because the dominant information technologies of the day control all understanding and its illusion. . . . What counts are not the messages or the content with which they equip so-called souls for the duration of a technological era, but rather (and in strict accordance with McLuhan) their circuits, the very schematism of perceptibility" (*Gramophone, Film, Typewriter* xl-xli). The circuits represent the materialization of software as thing, hardened programming, and memory hardened into storage. Yet we do not know what our writing does, especially now that it mixes into autonomous machine behavior. Elsewhere Kittler writes that “programming languages have eroded the monopoly of ordinary languages and grown into a new hierarchy of their own,” which he calls a postmodern Tower of Babel (“There is No Software” 148). It may be, as Ruskkoff argues, that there is enough disinterest among users for technology leaders to maintain their monopolies, perhaps because the insignificant people spend so much of their psychic energy manipulating user interfaces, what Ian Bogost mocks as clicking cows. The conclusion to be drawn on this front, in parallel to Bauerlein, is that the masses are one full dimensional leap behind those in power, releasing collective agency to machines along with elite human groups, for they are also not the ones who design what those in power manipulate effortlessly to their advantage. They are more like addicts, lucky they know how to operate them. Rushkoff's warning is that “before, failing meant surrendering our agency to a new elite. In a digital age, failure could mean relinquishing our nascent collective agency to the machines themselves" (20).

The Quintessential Postmodern Object

While the narrative I will articulate includes the aforementioned components leading up to the dumbest generation – the absent mind, mythinformation, closed world discourse, programmed visions – a subtle yet pervasive influence seems rooted in the different ways that people learn. This insight comes from the emphasis Lev Manovich places on Alan Kay's vision for the democratization of software development, with its roots in the latter's interpretation of Jerome Bruner's learning theories, themselves influenced by the work of Jean Piaget. Kay, like Kemeny and Engelbart, envisioned radical transformation of human intelligence through its collaborative, symbiotic interaction with computing machinery. He proposed a universal media machine, the *Dynabook*, for manipulating personal dynamic media, that, in a more real sense than Kemeny's teaching the computer to think by programming it, created a dialog between human and machine via its magical paper. Manovich connects Jerome Bruner's enactive, iconic, and symbolic mentalities to Kay's theorization of the optimal user interface featuring mouse, icons, and Smalltalk programming (*Software Takes Command* 97-98).

I see this as a crucial development. *Theorizing learning as having enactive, iconic and symbolic components means that removing the obligatory need to program from a command-line interface – the symbolic – may have unintentionally weakened human intelligence*. Kemeny mistook this redistribution of learning modes as the simplification of the interface, with the unintentional consequence that procedural rhetoric is no longer learned in the process of using computers, which he articulated in his criticism of computer-aided instruction. Taking this approach shunts the heated, politicized debates that revolve around arguments hinging on the absent mind, mythinformation, closed world, or programmed visions discourses, and replaces them with a more general point about how we relate to technology, how we *comport* ourselves.[7]

The history of personal computers is replete with narratives of how command-line, symbolic interfaces were superseded by graphical environments, GUI desktops and now touch sensitive interfaces—through means Bruce Schneiderman calls *direct manipulation*. A likely pattern develops when it comes to philosophical contemplation of, as David Brin puts it, "why Johnny can't code." For many years now, everyday computers do not come ready to learn programming – and most today do not, although online tutorials abound – and the consequences are ominous, for multiple generations have grown up without learning to write code or pick up a soldering iron. We could all fall into new dark ages if the global supply of capable technologists diminishes beyond a critical threshold, or the machines take over as dramatized in many science fictions. That is a limit case of which the dumbest generation represents the more likely outcome for the majority of humans.

Is it only historical accident, as Manovich suggests, that the Macintosh did not ship with a user development environment, corrupting Kay's vision? Without an accessible programming interface, there can be no instructive play like Seymour Papert wished for LOGO, and Brin implies the early personal computers provided. Clearly, access to programming takes a different

7 Rudy McDaniel pointed out in an early draft that counterarguments to each of these are easily made. The absent mind can be viewed as the human brain freed up for high order thinking as Leibniz originally envisioned; we don't really need to pay attention to the mundane. Beside mythinformation are changes in discourse that do not entail stupidity; new language patterns, e.g. memes, reflect evolution. Closed world discourses are countered by the global reach of the Internet, social media for used for justice, the Arab spring, etc. Finally, programmed visions can be viewed as new forms of control replacing prior, similarly repressive and overdetermining forces.

form today, and may be greatly diminished. There is no `READY>` prompt, as Montfort et. al. declare in their recent work from 2013 on critical code and platform studies, whose title is the famous one-liner program for the Commodore 64*, 10 PRINT CHR$(205.5+RND(1)); : GOTO 10*. Indeed, Sherry Turkle famously states that the Apple Macintosh is the quintessential postmodern object, and recasts the relationship between human and machine as conversational rather than dictatorial. “Unlike the personal computers that had come before, the Mac encouraged users to stay at a surface level of visual representation and gave no hint of inner mechanisms. . . . The tools of the modernist culture of calculation became layered underneath the experience of the culture of simulation” (*Life on the Screen* 34-35). We seem to be set on the trajectory of becoming *WALL-E* passengers of Spaceship Earth, characterized by entrained striations, choice filter creation, and programmed visions.

Foss Hopes

Free, open source software that emerged in conjunction with the proliferation of the Internet and World Wide Web gained popularity in philosophical and political debates during the early 2000s, and certainly it is much easier today to explore programming than it was in the heyday of the Microsoft monopoly over personal computer operating systems. Nonetheless, code, to Kittler, presents an insoluble dilemma yielding random buzz either way it is pursued, traditional stored-program or connectionist. “The so-called 'hidden layers' in today's neuronal networks present a good, if still trifling, example of how far computing procedures can stray from their design engineers, even if everything works out well in the end. Thus, either we write code that in the manner of natural constants reveals the determinations of the matter itself, but at the same time pay the price of millions of lines of code and billions of dollars for digital hardware; or else we leave the task up to the machines that derive code from their own environment, although we then cannot read – that is to say: articulate – this code. Ultimately, the dilemma between code and language seems insoluble" ("Code" in *Software Studies* 45-46).

Kittler's disheartening conclusion suggests that instead of studying code directly, we must be good scholars and cultural observers, spectators. For he judges the division between code and discourse insurmountable. The emergence of free, open source software (FOSS[8], henceforth *foss*) in the late 1990s onward seemed to provide avenues both to study hitherto inaccessible source code, and give individuals and ad hoc collaborations an advantage in technical invention by leveraging Internet resources without the overhead of institutional and corporate protocols.[9] While Richard Stallman, who created the GNU Public License (GPL), insists that the scarcity of willingness to work together for the public good, not scarcity of technical innovation, is the root evil of non-free software ("Why Software Should Be Free" 124), the fact that a generation or more has grown up learning to use applications and tinker with settings, instead of literally writing code, should be considered in the analysis of the gains and losses that humans and machines have undergone. While it may be excessive to contend that settling for this form of

8 Others insist on the acronym FLOSS to emphasize the free, as in libré, free speech versus free beer.

9 The essence of free software is captured in the four freedoms of the GNU Project begun by Richard Stallman: "You have the freedom to run the program, for any purpose. You have the freedom to modify the program to suit your needs. (To make this freedom effective in practice, you must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.) You have the freedom to redistribute copies, either gratis or for a fee. You have the freedom to distribute modified versions of the program, so that the community can benefit from your improvements" ("The GNU Project" 18).

comportment with our devices has contributed to making humanity collectively dumber, it feeds the science fiction narrative foreshadowing future generations of *WALL-E* consumers satisfied with the interface level of interaction.

Some misconceptions need to be dispelled, what I call *foss hopes*, as the putative revolutionary, democratizing, ennobling potential inherent in free, open source software, which seemed poised at the turn of the twenty-first century to reverse the negative dominance of closed, proprietary software systems that more than anything else demonstrate the truth behind the claims that digital media is big business. I and many others may have made what I call the *Theuth error*, in honor of the character from the Platonic myth, expecting more from foss as epistemologically enlightening, and economically and socially democratizing, empowering the little people towards the equity imagined by Lanier built up from lifetime network activity as some strange new kind of individual property, than we should have without a fundamental resurgence of programming education.

The initial exuberance heralded by works like Eric S. Raymond's *The Cathedral and the Bazaar*, Linus Torvald's *Just for Fun*, Pekka Himanen's *The Hacker Ethic*, and countless opinion pieces that followed the maturation of GNU/Linux and other foss alternatives to commercial applications from fringe to mainstream use has since been tempered by the findings of empirical research, for example Feller et. al.'s *Perspectives on Free and Open Source Software*, now nearly a decade old. Research shows that as they mature, free, open source development communities tend to adopt the useful behaviors of corporate norms, professional software business practices, and become more like them. Furthermore, ethnographic studies like Yuri Takhteyev's *Coding Places: Software Practices in a South American City* reveal entrenched Anglocentrism in both programming language design, actual code bases, and community discussion forums.

Moreover, central to Lanier's argument that siren servers have corrupted the Internet is the fact that foss made cheap networking possible, and the illusion that Internet services are free, as in 'free beer' versus 'free speech', conceals the social costs of the arrangements that have developed, like the hidden costs of our culture built around private automobiles that Ruskkoff criticizes. The freedom to browse the web, join social media sites, play games, and view pornography without ponying up any cash, with the near effortless gesture of moving a computer mouse, is like the freedom to drive public roadways to reach any point in the United States with the near effortless gestures of pressing the gas pedal and turning the steering wheel. At the same time, the hidden costs and constraints of the foss-enabled Internet may impact the environment, society, and individual lives in ways analogous to the long term effects of greenhouse gases, suburban sprawl, financial indenture, and socio-economic stratification of the automotive, petrochemical, and financial industries. Defaulting, we depend upon large corporations to provide and manage cyberspace, though I will suggest that we can home in on *default philosophies of computing*, by studying the writing, code, and engineering work of dominant technologists in the short modern history of computing.

Default Philosophies of Computing

Bruno Latour exclaims that “there is no greater intellectual crime than to address with the equipment of an older period the challenges of the present one” (“Why has Critique Run out of Steam?” 231). The postmoderns are the very ones who talk about their Macintoshes while still writing books; their hearts are in the right place but minds focused on the previous technological era. At the same time, there is at this border a place for thinking the most powerful and the most thoughtful do not enter, that system designers and programmers regularly explore. The smooth operation of German railways, the political decision to use atomic bombs, and Father Roberto Busa's *Index Thomisticus*, were supplied by early versions of technologies that Bill Gates sees on the horizon in his 1995 book *The Road Ahead*, the same ones that are under intense scrutiny today following the revelations by Edwin Snowden over NSA data collection practices. Gates wrote that “it might take only a few more incidents like the bombing in Oklahoma City within the borders of the United States for attitudes toward strong privacy protection to shift. What today seems like digital Big Brother might one day become the norm if the alternative is being left to the mercy of terrorists and criminals. I am not advocating either position – technology will enable society to make a political decision” (269-270). Gates' claim that technology will foster political decision making begs the question, *how exactly do we philosophize about computing?* He thinks from his incredibly privileged perspective, yet it is still a human perspective mediated by the computing machinery of the time.

Academic debate inspired by a philosophy of computing tradition is still unconscious of its preference formation; there is no explicit philosophy of computing tradition yet, although there are philosophies of information, semiotics, software, computer ethics, and so on. In this absence of a clear directive for philosophy to address computing, the void fills by default: futurists, technophobes, politicians, evangelists, industry leaders, portrayals in science fiction, and with the ideologies concretized in extant technological systems themselves. As a recent example, the 2014 film *Transcendence* dramatizes two long-standing dreams of artificial intelligence, the emergence of global machine intelligence far exceeding human ken, referred to as 'the singularity', and the duplication of human consciousness in those same systems. These are precisely the themes Ray Kurzweil promotes in his 1999 book *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*, with the goal, he states, of enhancing predictions focusing on demographic, economic and political trends with emerging machine capabilities as intelligent agents (10).

While we may smirk at Kurzweil's enthusiastic, often extreme rhetoric, he gives himself the license to prophesize because he has spent his career successfully developing innovative computer systems that perform, among other things, speech recognition, one of the early holy grails of the human-computer symbiosis identified by Licklider. Gates, for a long time the wealthiest human being alive, also ranks among those I am calling default philosophers of computing because, as both inventor of BASIC for eight-bit personal computers, and whose handiwork at the head of Microsoft touched millions of machines and people. Using a travel guide metaphor for the information superhighway he was building insinuates an assessment of the opinions of everyday consumers as having little importance beyond accepting the technologies that have been designed and marketed to them, then heavily influenced by marketing.[10] Trickle down prosperity might be described as an underlying philosophical position of Gates; shifting richness defining the good life is first enjoyed by the few, then served up to the masses. The analogy to the physical highway system Rushkoff decries as having developed through ordinary citizens' ignorance of the workings of the automobile industry is palpable.

10 For many years during the backlash against its monopolistic practices, sarcastic versions of its trademarked slogan “Where do you want to go today?” proliferated. “Where does Microsoft want to drag you today?” becomes the response to years of its perceived heavy-handed shaping the technology landscape.

Digital Humanities Solutions

Gates believes the availability of information will spark curiosity, whereas Bauerlein argues that unguided and uninformed by tradition, children are lured into weaving themselves a cocoon of limited peer interests with their personal digital devices. Nonetheless, Gates embodies and enacts his philosophy of computing, while Bauerlein and Johnson merely comment about it. Producer versus consumer experience with technology gives his words a sort of ontological precedence. If we do not want to be dragged into our relationships with technology, then we must surpass these default philosophies of computing. How can scholarship engage without falling into the same traps plaguing sayers rather than doers? Digital humanities solutions present themselves as novel ways to approach this goal.

Not to Use Old Tools for New Problems

According to Ian Thomson's reading of Heidegger, America has been the avant-garde of *ontohistorical technologization*, that is, our nation is the one working the hardest to obscure the insight that we humans are *not* entities making ourselves, in an odd reversal of the Socratic imperative to know thyself (155). Philosophers of technology in the hermeneutic, phenomenological tradition that follows Heidegger often see their task as curing this delusion. Yet Heidegger himself has been criticized for the apparent indifference that is a consequence of conceiving the essence of freedom at such a high level. Similar questions may be posed when encountering the self-assured predictions – and personally-backed corporate missions – of Kurzweil, Gates, and other default philosophers of computing introduced in the previous section.

Even on our home turf, Walter Ong, a founder of digital humanities, suggested that programming languages are not like natural, spoken ones. Is there a loss in philosophical space resulting from rejection of computer languages?[11] Even Deleuze and Guattari, whose anti-establishment, rhizomatic, bricoleur meanderings seem to play into the bazaar development model championed by free, open source advocates, appear to dismiss the need for empirical verifications of philosophical concepts that could be accomplished through engineered solutions (*What is Philosophy?* 33-34).[12] Deborah Johnson's summary deprivileging of the standard account of computer ethics in the fourth edition of *Computer Ethics*, and introduction of the sociotechnical systems perspective, represents the attempt to deploy a new tool. However, splicing in the perspective of a computer scientist leaves a fundamental methodological gap because her insights do not themselves arise from long time immersion as a practicing theorist.

11 Ong's unconscious is revealed in his prejudice about the possibility of computer languages ever emerging from the unconscious like a mother tongue. Does Ong imply that nobody would ever write dreamy software that borders on being sensible to both humans and machines? Or does Ong imply that computer languages, as the extreme case of learned languages, while they may be relevant or related to written texts, have nothing to do with orality? What about computer systems that know common, spoken and written languages like English, written languages like ancient Greek and Latin, as well as C, C++, Perl, PHP, HTML, HTTP, and so on and so on? Why can't they learn to speak? How is electronic culture distorted by thinkers bound to print and orality? Really we are continuing Ong's analysis, focusing it upon itself, that is, print culture's prejudices distorting the possibilities and ontology of electronic culture.

12 There are contingents within the AI and cognitive science communities who do extensively employ simulation and modeling to test or illustrate various philosophical concepts, including Selmer Bringsjord, Robert Cavalier, Patrick Grimm, and other members of the International Association of Philosophy and Computing.

Mainstream humanities have also been popular targets for failing to adapt their practices. As Jeff Rice reveals, composition research has always been wary of using new tools, whether the typewriter or computer display, without prior, in-depth empirical study; meanwhile generations grow up using these tools daily, developing practices that outstrip research (143-144). G. Thomas Tanselle argues the computer as tool does not fundamentally alter reading or subjectivity (3), whereas Manovich, Hayles and others strongly disagree. The dismissive perspective seems to not consider digitally native electronic works, only electronic versions of texts originally composed with prior media forms. The intermingling of discourse systems and dynamic mechanisms generating *real virtualities* force traditional, print-oriented humanities out of this obsession with conceptions rooted solely in natural languages (Castells xxxi). In his manifesto for new tools and practices for humanities scholarship, *E-Crit*, Marcel O'Gorman describes the failure of theory in applying deconstruction to revolutionary scholarly practices: “somewhere in the early 1990s, the major tenets of deconstruction (death of the Author, intertextuality, etc.) were displaced into technology, that is, hypertext. Or to put it another way, philosophy was transformed, liquidated even, into the materiality of new media” (xv). The outcomes he notes continue to replicate traditional, print-based scholarly practices.

Scholarship Requires a Cybersage

In a curious twist near the conclusion of *The Dumbest Generation*, the author blames the culture war instigated by the 1960s New Left for initiating the overall decline of intellectual life by rejecting reading and learning obsolete and irrelevant topics (226). This accusation relates to a dilemma I find at the heart of the philosophy of computing: one the one hand, to understand technology, one must learn to use it, yet the state of the art is so complex that it is unmanageable for novices; on the other hand, beginning with simpler, older technologies that Brin recommends is a non-starter for those who adhere to what I call the *ponenda non sumeret* tenet, to not waste one's time dabbling in obsolete, useless topics.[13] The unfortunate outcome is that ignorance of technical details shunts formation of places for philosophical thought to occur, such as in the working code of critical programming studies.

Michael Heim made what I call the cybersage declaration for addressing the metaphysical sphinx of computer technology epitomizing the all-enframing Gestell proposed by the now deprecated Heidegger: “the Schreibstübe is giving way to the computer workstation, and scholarship requires a cybersage" (“Computer as Component” 304-305). John von Neumann, hands-on cybersage of early electronic computing, made a statement that will haunt cybernetics and the humanities philosophy of technology for the following decades, suggesting the Socratic command to know thyself ought to be addressed through studying technology. His version claims "of all automata of high complexity, computing machines are the ones which we have the best chance of understanding. In the case of computing machines the complications can be very high, and yet they pertain to an object which is primarily mathematical and which we understand better than we understand most natural objects" (“Complicated Automata” 435). We sense a

13 Referring to a passage from a letter of Seneca titled “Philosophy and Progress” where he states “No, the sage did not withdraw from the mechanic arts, as Posidonius thinks, but never touched them at all. The sage would never have esteemed an invention worth making if it was not likely to merit permanent use; he would not have taken up what would have to be laid aside” (234).

revised perspective for the relation of human subjectivity to the environment, shifting a portion of the active, cognitive burden to the distributed symbiosis with the machinic.

To date there has been little serious academic philosophical or practical appraisal of the emergence of technological unconsciousness of machine-readable and coded objects for everyday life (Kitchin and Dodge 61). Nonetheless, how we compute shapes how we think, echoing Nietzsche and Kittler, and ultimately what we are. To notice any of this entails appreciating the importance of 'moments of plasticity' through social organization resulting in crystallization of particular techniques and technologies, well described by Jonathan Sterne in *Audible Past*, applied to computers, networks, software. As Bruno Latour argues, “the itinerary of facts becomes as easy to follow as that of railways or telephones, thanks to the materialization of the spirit that thinking machines and computers allow. . . . Reason today has more in common with a cable television network than with Platonic ideas” (*We Have Never Been Modern* 119).

There is value in studying technology, especially thinking machines, to better understand epistemology thanks to this materialization of spirit. Turkle, back in 1984, notes the problem of the novelty wearing off to the point that culturally poignant observations about computers disappear into the background like their disappearing interfaces; “our culture will develop ways of thinking about the computer that, in a sense, require no thought” (*Second Self* 331). The concealment of being in the non-thinking comportment to technology foreshadowed here repeats the ontotheological threat that kept Heidegger in his mountain hut and perpetuated the antiquated image of the thinker. The cybersage scholar I am theorizing here does not turn away, but instead engages in digital humanities projects that meet this dangerous, spreading indifference head on.

Digital Humanities Projects

A key feature of academic scholarship falling under the general term *digital humanities*, besides leaning heavily on technological components in its methodologies, is adherence to rigor and systematic, unambiguous procedural knowledge characteristic of the sciences, applied to humanities problems previously treated serendipitously through narratives and literary associations (Hockey 3). In a sometimes critical, reflexive fashion, the scope of humanities questioning has grown to include the very software development and data collection techniques employed as its tools. Busa epitomizes the first digital humanities period that Hockey presents, long before personal computers and the Internet. Next the personal computer period of mid 1980s to early 1990s freed humanities computing from the computing centers, their expertise, and scrutiny. The result was much duplication of effort, but also innovation, which I find comparable to the cathedral versus bazaar models of software development popular in foss discourse networks. Textuality, traditional archival and editing projects have reigned from the earliest days to the era of personal computers and into the Internet era.[14]

Marie-Laure recommends do-it-yourself (DIY) genres of democratized art such as Ulmerian artifacts. I suggest with the DIY focus blending in technical skill exercises and meditations on machine and posthuman embodiment, and at the same time, enter new areas beyond textuality, for instance the audible, via software synthesized sound. Foss projects become

14 Hockey notes the Orlando Project, a forerunner of recent instances like Hayle's *Electronic Literature* attempt to establish as classes of new creations. Poundstone's *Project for Tachistoscope*, for example, enacts high speed process control system cognition done by machinic others.

quintessential digital humanities projects as multipurposive, authochthonous, democratic, amateur, distributed network phenomena.

The excitement of theorizing about foss and the GPL peaked during its newness phase and has subsided, diminishing domains of scholarship referencing computer program source code. Digital humanities projects do not yet explicitly focus on their programming, the way composition studies and writing about writing have become an established subdiscipline. During the second decade of the twenty-first century a few groups have emerged. “Key to Critical Code Studies will be the development in practitioners of programming literacy,” writes Mark Marino in this group's formative statement (np). Heim declared the word processor was the calculator of the humanist, but it has already been revealed that the becoming calculator of first generation digital humanities projects represented a shunting of philosophical questioning away from the act of programming. Crucially, and dissolving all differences within the electronic era, *using computers and networks is different than using calculators* because we barely know what we are asking them to do, as we manipulate the GUI, and hardly teaching them how to do it, as Kemeny insisted was the right way to combine programming and arithmetic.

Rushkoff elegantly describes this absent mind pursuing its interests through distracted clicking via devices we engage fetishistically. “With computers and networks, unlike our calculators, we don't even know what we are asking our machines to do, much less how they are going to go about doing it. Every Google search is at least for most of us a Hail Mary pass into the datasphere, requesting something from an opaque black box” (23). Understanding biases through their embedded 'procedural rhetorics' is the guiding philosophy for getting on top of the problem posed by rapidly transforming technologies that seem to have taken command on their own, and the method to do it I call critical programming.

Critical Programming Studies

Theorists of literature, composition and media studies who align with digital humanities have made many contributions to what could be called philosophies of computing technology, although I prefer to consider these efforts preparatory while the discipline interpellates itself as such. George P. Landow credits Gregory Ulmer with establishing grounding philosophical arguments connecting hypertext theory to Derrida and other poststructuralist and postmodern thinkers (344-345). Yet Landow's blind spot, similar to Johnson's and so many others', surrounds the topics near and dear to technologists who are, in the words of Ellen Ullman, *close to the machine*. I, coming from the technology ranks, will present the programming perspective to complement Comparative Media Studies, and at the same time imagine a different trajectory of the unrealized potential in Landow, Turkle, and others, if there had been a generation of foss-equipped programmers instead of the dumbest one.

According to Chun, philosophy is just beginning to note effects of software as thing on metaphysics, intellectual property, subjectivity, and information (*Programmed Visions* 5-6). Ulmer gave digital humanities the term 'electracy' to name the period following orality and literacy, playing on the combination the traces so important to Derrida with electricity, electric writing. Ever since, however, critical theory has foregrounded the *traces* rather than the *electrons*. The distinction I wish to make foregrounds computer programming where traditional digital humanities theorists continue to emphasize the traces, albeit now shimmering signifiers, passing through Derrida and other popular postmodern theorists' discourse networks. It is to acknowledge that the Big Other constituting collective human machine intelligence has a mind of its own, and we are obliged, as philosophers of computing, to seek to understand it.

Nigel Thrift employs Patricia Clough's term *technological unconscious* as the pre-personal substrate of conventions of address, the bending of bodies that underlies cognition, perception and movement (“Remembering the Technological Unconsious” 176). Infrastructure must be performative to become reliably repetitive; once it has, it takes on an active if little noticed role in structuring human experience. Kitchin and Dodge investigate how software generates new kinds of space and invests the mundane with new capacities of control and surveillance. Crucially, their subsequent list of reasons for why there has been little resistance to digital technologies does not include lack of general programming knowledge, making a huge opening for performing philosophical investigations of this condition through critical programming studies. As if in response to them, Rushkoff offers ten commands for the digital age to balance recognized biases of digital media, of which the most important is to program or be programmed (8). The cognitive competencies described by Ian Bogost, Nick Montfort, Michael Mateas, and others as 'procedural rhetoric' and 'procedural literacy' employ programming as a paradigmatic practice for coming to understand how complex systems operate through the distributed coordination of interconnected levels or layers acting together, which I call *diachrony in synchrony* and will develop as a key component of my critical methodology in chapter three.

To make the transition from using software in digital humanities research to its *critical* use is to make the software development life cycle an integral part of the iterative, dialectical process of thinking through humanities questions. Luc Boltanksi and Eve Chiapello, authors of the monumental work *The New Spirit of Capitalism*, acknowledge the roles played by the Prospero@ software application, its inventors, and the human preparation of management texts into data files so they could be processed by the software (xxix). Yet they do not ponder the influence of this effort on the development of their research.

Jerome McGann, on the contrary, views creating software solutions to pursue humanities scholarship with incessant reflection on their design processes, what he calls "poiesis-as-theory" (83). Whereas first generation digital humanists like Father Busa let their predefined philosophical vision guide their development and use of computing technologies, and Boltanski and Chiapello take advantage of but do not reflexively consider, McGann belongs to the self-reflexive, cybersage generation. Others like J. D. Applen and Rudy McDaniel, by providing a tutorial-like, detailed introduction to XML in the context of deploying digital rhetoric, mark the push for humanities scholarship towards technical competence, beginning with differentiation between HTML and XML. I feel that humanists and philosophers are well positioned to make the leap into what Applen and McDaniel call *theorist-practitioner* roles, undertaking critical programming studies forming not just disjointed projects but digital humanities solutions.

Following this introduction, the next two chapters will bridge the gap between the as-is situation – the legacy of the dumbest generation – and the projective trajectory suggested by McGann, Hayles, Bogost, Montfort, O'Gorman, Applen and McDaniel, and others who promote a new tools, theorist-practitioner paradigm deeply mixed with critical reflection upon the use and development of those tools. Future work will seek to advance digital humanities scholarship toward a philosophy of computing, by territorializing the little explored discourse networks of the *philosophical programmers* who are credited with developing the machinery, languages, network protocols, operating systems, and applications of the post literacy epoch. Digital humanities scholarship has the aim of changing the trajectory of the dumbest generation that has formed in its wake, by promoting foss projects that conduct critical programming studies, whose practitioners should be called *programming philosophers*. The command issued by Latour not to use old tools for new problems welcomes computer programming components into academic discourse networks through digital humanities projects. Forming the synthesis portion of this dissertation in chapter four, I will offer my foss triad `symposia`, `tapoc`, `pmrek` as examples of critical programming studies that intentionally parallel orality, literacy, and electracy.

The `symposia` project arose from my earlier studies in ancient Greek philosophy, inspired by imagining Plato's *Symposium* as the transcription of a home video recording of an actual event. For centuries, philosophers have been reading this text rather than listening to it. Concurrent text to speech processes pronounce the Socratic dialogue as if it was spoken by ten speakers arranged as in the implied virtual setting of the story. The `tapoc` (toward a philosophy of computing) project is literally the computational basis of this dissertation document itself, an amalgam of C++, Perl, PHP, and shell scripts interacting with the Apache webserver and MySQL database. It takes literacy to its limit as a combination of human and machine collaboration, and introduces – with a nod to Ulmer and O'Gorman – the philosophical concept of *fossification*, in which intellectual property is made permanently free by turning it into source code published under the GPL. Finally, the Pinball Machine Reverse Engineering Kit (`pmrek`) explores the boundary between human and machine cognition and embodiment by providing an experimental platform for building electronic circuits and writing computer code to control high speed, digital process control systems. In each project, acts of working code reflexively inform contemplation of philosophical subjects while concurrently developing practical skills in software development and electronic engineering.

CHAPTER 2: POST-POSTMODERN NETWORK DIVIDUAL CYBORG

The aim of this chapter is to survey what I am calling the as-is situation for human beings and machine technologies in the United States at the turn of the twenty-first century. In terms of rhetorical effect, few critical works addressing the human situation with respect to technological media more compellingly cast the serious need to study it than the opening paragraphs of the preface of Friedrich Kittler's *Gramophone, Film, Typewriter*. "Media determine our situation, which in spite or because of it deserves a description. Indeed: in 1941, with the knowledge of files and technologies, enemy positions and deployment plans, and located at the center of the Army High Command in Berlin's Bendlerstrasse, it may still have been possible to take stock of the situation" (xxxix). Imagine being there, trying to be in command amid overwhelming stacks of files and reports during the second World War. Kittler continues: “the present situation is more obscure. . . . [E]ven secret files suffer a loss of power when real streams of data, bypassing writing and writers, turn out merely to be unreadable series of numbers circulating between networked computers. Technologies that not only subvert writing, but engulf it and carry it off along with so-called Man, render their own description impossible. Increasingly, data flows once confined to books and later to records and files are disappearing into black holes and boxes that, as artificial intelligence, are bidding us farewell on their way to nameless high commands” (xxxix-xl). Yet now the situation ought to be more readily understandable, for its being modulated by decades of psychological research and studies of human computer interaction, shaped by the tendency to recast everything in rationalized terms amenable to modeling and testable circuits. I propose that there is value in teasing out a more detailed theory of what happened to the mutually augmenting human computer symbiosis over the course of the decades since World War II, such that its trajectory has veered towards diminished human capacity to thoughtfully interact with machines, permitting an equilibrium to exist in which the latter continue to become smarter, and the former continue to become less mobile, less industrious, more striated – *dumber*, as Bauerlein puts it.[15]

This chapter surveys the relationships between modernism, postmodernism, subjectivity, cybernetics, embodiment, and techno-capitalist networks, collectively constituting human/machine symbiotic being to produce an initial response to Kittler's challenge that we assess the current situation as actors steeped in the technological media specific to our sociohistorical milieu. The current intellectual climate in the United States – what I call the 'as-is situation' – taking off from the collective intelligence problems suggested by the rhetoric of the dumbest generation, passing forcefully through the technological era of early electronic

15 At the risk of putting off my human readers, I stick with this contention and challenge you to consider two senses of becoming dumber along side computer technology in spite of increasing ubiquity of human-machine interaction and physical and virtual spaces dependent upon code. First, as in dumbing down, simplifying, making more shallow the cultural exchanges and especially moral reasoning that Bauerlein and Turkle argue have been diminished on account of American youth interacting almost exclusively in their limited social media networks, no longer engaging with tradition nor finding adult relationships relevant. Second, as in dumber, unable to speak intelligently with computing machinery, due to the failure of general programming education of the sort Kemeny imagined taking hold. Instead, we have computer literacy that is restricted to the surface, interface level, limited how-to knowledge, as Rushkoff reveals.

computing into the Internet epoch, I theorize as that of the *post-postmodern network dividual cyborg*.[16]

Michael Hardt and Antonio Negri do an excellent job articulating the transition from modernity to postmodernity, and compellingly argue that the present age goes beyond postmodernity into what they call, in their book of the same name, *Empire*. The particular mode of subjectivity that has developed in the western world, particularly the United States, appears regressive when measured against the modernist ideal. By calling it Heidegger's America, I appeal to theorists like Benjamin, Horkheimer, Adorno, Barthes, Baudrillard, and Derrida, who carry on Heidegger's work by joining their critiques of machine technology with the flaws of capitalism. At the same time, I acknowledge the conclusion of Hardt and Negri, seconded by Boltanksi and Chiapello, that a new, autonomous, global order of biopower or general intelligence has taken control away from the plans of individuals as well as from transcendent religious sources, so that it seems to have a mind of its own, casting the majority as the consumer herd to be shepherded toward a metastasis of *WALL-E* somnambulism.

In societies of control in the epoch of electracy, humans and machines enter complex networks and ways of being qualitatively different than from those disciplinary regimes consummating the epoch of literacy, for the latter are bounded by the limits of natural automata and mechanical media. In deeply commingling technogenesis and synaptogenesis, cybernetics combine with embodied life along not just intellectual but also affective, visceral dimensions. It

16 The inspiration for developing the term 'post-postmodern' is Richard Johnson's coining the term 'post-post-structuralist' in his account of subjectivity emphasizing the self-production of subjects in “What is Cultural Studies Anyway?”.

is not surprising that popular threads of continental philosophy merge with science fiction narratives about posthuman cyborgs. Human being as *dividual*, rather than individual, operates in these spaces. With full appreciation of the technological component, it becomes the dividual cyborg.

Post-Postmodern Subjectivity

The liberal humanist subject signifies the predominantly cognitive, intellectual, spiritual individuality of a specific, putatively universal biological creature, yet has been demonstrated to reflect not only ocularcentric biases specific to print media, but also the preferences of hegemonic male, privileged actors. The emblematic expression of this confident subjectivity, explored by Foucault through the *Las Meninas* painting by Velasquez that begins *The Order of Things*, illustrates the modernity appeal to sound metanarratives including geometrical perspective and lighting, and hence is emblematic of the inauguration of great periodization theories that mark the modern with Enlightenment ideals. Following a pattern well established in the digital humanities of enframing fundamental philosophical positions within their technological age, the constitution of the broad notion of human being is formed around the physical, cultural, and technical contours of their dominant media technologies.

In this broad periodization model, centuries of sway by print literacy begin to be surpassed by dynamic, electrically powered media in the late nineteenth through mid twentieth centuries, ending the period generally called modernism and ushering in the period after, the post modern, whose conception of humanity was threatened by dissolution of guiding metanarratives into inauthentic relativisms featuring the precession of simulacra in unconscious market economies. As electronic control surpassed electromechanical regulation, a new model of the soul emerged that was far superior to the gramophone, film and typewriter construction Kittler develops: the automatic digital computer. At the same time, the unified, private individuality of the modernist subject, splintered into the schizoid, postmodern dividual, begins to reassemble around a virtual core whose sense organs, memory and cognitive faculties extend into the inhuman environment. However, as convincingly argued by Catherine Malabou in *What Should We Do With Our Brain?*, it has become difficult to distinguish between the latest findings of neural science and the prevalent rhetorics of global capitalism. The fate of the dividual cyborg therefore seems to be at the mercy of transnational corporate agendas that manufacture desires and means to achieve them, consummating the regressive consumer mentality railed against by Frankfurt school critical theorists like Horkheimer and Adorno.

Modernism and Postmodernism

In intellectual histories of the Western world, the modernist period stretches from the Renaissance and Enlightenment through the early twentieth century. It is recognized by the affirmation of autonomy against traditional authorities, the internalization of discipline, and the appeal to a legitimating grand narrative or metadiscourse (Feenberg *Transforming Technology* 162; Turkle *Inner History of Devices* 135; Lyotard *Postmodern Condition* xxiii). It is also coextensive with the birth of print humanities and widespread literacy, made possible by the mass production of texts (Misa 25). Latour lays out the modern constitutional guarantees in which nature and the social are constructed but seem natural, distinct from prior mediation, under the sway of "God now crossed out" (*We Have Never Been Modern* 32).

The process of modernization internalizes the outside, but to Latour, the modernist project never completes itself, leaving hybrids proliferating at the boundaries. Hardt and Negri likewise demonstrate that modernity is defined by “a crisis that is born of the uninterrupted conflict between the immanent, constructive forces and the transcendent power aimed at restoring order” (76). That is, the emancipating influence of scientific reason over religious teleology and democratic republicanism over patrimonial monarchy generated counter effects in which entrenched powers shifted their hegemony to nation-states and capitalist markets, while continuing to impose mediations between everyday human life and the immanent environment. Thus for Hardt and Negri, Hobbes' social contract defines sovereignty by transcendence and representation, and Rousseau's republican absolute is equivalent to Hobbes's God on earth, then pointing to capitalist markets as the foundation of values as articulated by Adam Smith. Modernity transfers spiritual identity to nation-states, and subjects become citizens (95).

Hardt and Negri argue there is an intimate relation between the crisis of modernity to racial subordination and colonization; the nation-state is a machine producing Others (114). Indeed, colonial slavery that was key to European commerce and the process of capital development depended on external populations of Others to suppress, and in many cases internalize. Under modernism, critical theory and Maxism arise as putative alternatives to the positivist use of knowledge to regulate society, but were still used to program the system, and assumes society is a giant machine (Lyotard *Postmodern Condition* 12-13). The utopias imagined by Enlightenment humanists failed to actualize, co-opted by nation-states and capital, and we got the modern world instead.[17] Thus they argue that even Foucault's theoretical methodology is not

17 I make this point to parallel similar hopes for bright futures wedding humans and computers in symbiosis.

different from the modernist Enlightenment mandate epitomized in Kant: “*Sapere aude* (dare to know), emerge from the present state of immaturity, and celebrate the public use of reason at the center of the social realm. Foucault's version, when we situate it historically, is not really all that different. . . . In this ebb and flow between inside and outside, the critique of modernity does not finally go beyond its terms and limits, but rather stands poised on its boundaries" (183-184).

The Postmodernist thought of Lyotard, Baudrillard, and Derrida, among others, challenges binary logics of modernity (Hardt and Negri 139). Lyotard, who coined the term, applies it to “the condition of knowledge in the most highly developed societies. . . it designates the state of our culture following the transformations which, since the end of the nineteenth century, have altered the game rules for science, literature, and the arts” (xxiii). Crisis in the narratives supporting in the modernist position, from the legitimacy of scientific thought to democratic freedom and Eurocentrism, mark the postmodern condition. Frederic Jameson claims that “it is safest to grasp the concept of the postmodern as an attempt to think the present historically in an age that has forgotten how to think historically in the first place,” and “may then amount to not much more than theorizing its own condition of possibility, which consists primarily in the sheer enumeration of changes and modifications” (ix-x).

It is not surprising that a common complaint about postmodernism is its absence of sustained theory in the first place. Lyotard claims postmodern works without rules until the work is complete enough that they appear after the fact (81). Continuing Jameson's description, “in postmodern culture, culture has become a product in its own right; the market has become a substitute for itself and fully as much a commodity as any of the items it includes within itself:

modernism was still minimally and tendentially the critique of the commodity and the effort to make it transcend itself. Postmodernism is the consumption of sheer commodification as a process. The life-style of the superstate therefore stands in relationship to Marx's fetishism of commodities as the most advanced monotheisms to primitive animisms or the most rudimentary idol worship" (ix-x). As a general situation of temporal disjunction, the postmodern process makes everything artificial, throwing sustaining master narratives into disarray.

Postmodern artificiality is not a poor modeling of reality, for the real has merged with representation in the immanence carried over from modernity. Of the three orders of simulacra—counterfeit, production, and simulation—the third, simulation, carries the day. As Baudrillard puts it, “simulation is no longer that of a territory, a referential being, or a substance. It is the generation by models of a real without origin or reality; a hyperreal. The territory no longer precedes the map, nor does it survive it. It is nevertheless the map that precedes the territory – *precession of simulacra* that engenders the territory" (*Simulacra and Simulation* 1). It is worth noting that Baudrillard associates each order of simulacra with a corresponding 'law of value': natural, commercial, and structural (*Simulations* 83). A structural law of value is the only type that remains in the homogenization of digital data; thus it is not surprising that Turkle and others associate computer technologies, which elevate the structural to ontological primacy, with postmodernism.

Accompanying the hyperreal is the victory of systemic nihilism, the third type after its aesthetic and metaphysical forms. Baudrillard declares “death no longer has a stage, neither phantasmatic nor political, on which to represent itself, to play itself out, either a ceremonial or a violent one. And this is the victory of the other nihilism, of the other terrorism, that of the system" (*Simulacra and Simulation* 164). Hardt and Negri consider the present condition to be the omni-crisis of postmodernity because “the binaries that defined modern conflict have become blurred. The Other that might delimit a modern sovereign Self has become fractured and indistinct, and there is no longer an outside that can bound the place of sovereignty” (189). It is fitting, then, that Walter Benjamin declared the necessity of war to preserve the property system and belief in rampant scarcity, and Lyotard found Horkheimer's paranoia of reason caught in a feedback loop with the belief that society is a giant machine, a technocrat unicity (*Postmodern Condition* 12).

Particularly significant to the transformation of the liberal humanist subject is the postmodern abandonment of foundational grand narratives supporting its beliefs. Lyotard prefers to cast thought in terms of language games, for which narrative forms are appropriate but not hegemonic (*Postmodern Condition* 18-19). As he puts it, “we no longer have recourse to the grand narratives we can resort neither to the dialectic of Spirit nor even to the emancipation of humanity as a validation for postmodern scientific discourse. But as we have just seen, the little narrative [*petit recit*] remains the quintessential form of imaginative invention, most particularly in science. . . . It is the object of administrative procedures, in Luhmann's sense” (60-61). Moreover, he contends that these systems rely on the arrogance of their decision makers to terrorize other players with the threat of exclusion from the language games to consolidate their power (63-64). Thus, according to Jameson, language overdetermines thought and activity in the manner of Foucault's panopticon. “This is the sense in which, even if Big Brother is not everywhere watching you, Language is: media and specialized or expert language that seeks tirelessly to classify and categorize, to transform the individual into the labeled group, and to constrict and expel the last spaces for what was in Wittgenstein or Heidegger, in existentialism or in traditional individualism, the unique and the unnameable, the mystical private property of the ineffable and the unspeakable horror of the incomparable" (322). People intrinsically realize their knowledge is legitimated only by linguistic practices and communicational interaction. As Baudrillard puts it, "the stage of analysis itself has become uncertain, aleatory: theories float" (*Simulacra and Simulation* 161). Computer technologies, especially network protocols, epitomize this reifying character of language games that have come to form the quasi-material basis of the built environment and the patterns of information exchange among machinery.

Lyotard has the premonition that all research is dictated by subsumption of results as computable information, driven by an ideology of communicational transparency (*Postmodern Condition* 5). Postmodern mourning the loss of meaning reflects the spread of scientific knowledge at the expense of narrative knowledge. His own solution to the postmodern condition is in *paralogy*, movement against these established ways of reasoning based on terrorist language games. "Paralogy must be distinguished from innovation: the latter is under the command of the system, or at least used by it to improve its efficiency; the former is a move (the importance of which is often not recognized until later) played in the pragmatics of knowledge" (*Postmodern Condition* 61). He favors open systems, especially in science, for their ability to generate new ideas, alternate statements and game rules from the prevailing ones. However, this counter position of modernist rationality that emphasizes communicational transparency at the same time shrouds its depths.

As Turkle explains in *Life on the Screen* in the context of popular computer games, “the notion of worlds without origins is close to the postmodern challenge to the traditional epistemologies of depth. . . . If there is no underlying meaning, or a meaning we shall never know, postmodern theorists argue that the privileged way of knowing can only be through an exploration of surfaces. . . . The French anthropologist Claude Levi-Strauss described the process of theoretical tinkering – *bricolage* – by which individuals and cultures use the objects around them to develop and assimilate ideas” (47). Despite the positive spin often associated with this term, bricolage thinking, when combined with a world populated by ready-made, mass-produced consumer goods, points toward the formation of a regressive subjectivity. Indeed, Jameson suggests that “the postmodern may well in that sense be little more than a transitional period between two stages of capitalism, in which the earlier forms of the economic are in the process of being restructured on a global scale, including the older forms of labor and its traditional organizational institutions and concepts" (417). Nonetheless, the impact on human being – consciousness, subjectivity, intelligence, 'so-called souls' per Kittler – of the large scale cultural transformations that are associated with the postmodern period are generally deemed negative.

Regressive Subjectivity

The progression to grasp in this section is first the question, what is the self and subjectivity in western culture?, and second, how did it become regressive? Eric Havelock, Walter Ong, and many others credit Plato for helping to invent the self out of the Socratic dialogue, though Kittler finds the widespread adoption of writing and reading over many many generations a more fitting explanatory source. Havelock singles out the practice of reading as key. In *The Muse Learns to Write*, he poses the question, “why choose vision as the metaphor for an intellectual operation, unless guided by the subconscious recognition that the operation had arisen out of viewing the written word rather than just hearing it spoken? . . . The self was a Socratic discovery or, perhaps we should say, an invention of the Socratic vocabulary" (111). Jasper Neel reminds us in *Plato Derrida Writing* that nobody believes that Plato's texts are verbatim accounts of extemporaneous oral events. “No one thinks the sort of structure demanded by Socrates' dictum could be achieved in speech. . . . Plato was creating thinking, and the only way he could create the kind of thinking he did was in writing, because the sort of disinterested speculation advocated throughout the Platonic canon is impossible in a preliterate culture” (42-43). According to Micheal Heim, who was heavily influenced by both Havleock and Ong, literacy produces literate minds with psychic qualities of wholeness of attention, contemplative presence, and distance from mundane pressures. Their shared hypothesis is that individualized, private, reflective, conscious subjectivity is qualitatively distinct from humans immersed in the immediacy of oral cultures.

Importantly, Heim notes an interpretive distortion has occurred, urging that the progression from an oral to a literate subject should not be credited to new skills in information handling and technical apparatus alone, but instead “the activity of forming, of ideational focus, belongs to the kind of insight fostered by the book. . . . The value of literacy, in the Platonic tradition, resides in the fact that literacy produces literate minds. Far from tautological, the notion of a literate mind includes psychic qualities such as wholeness of attention, contemplative presence of mind, and a distance form the mundane pressures which scatter and fragment human experience” (*Electric Language* 185-186). Heim argues further that chirographic (handwriting) culture dissociates knowledge from the speaker, while retaining situated, vocal presence, whereas with the ascendancy of typography, modern logic displaced many of the rhetorical structures employed by oral reasoning to form a decontextualized, aloof thinking subject as theorized by Descartes leading into the modernist tradition (62-63).

A significant portion of Derrida's *Of Grammatology* is spent connecting writing to the emergence of subjectivity. “Under the name of writing, Condillac thinks readily of the possibility of such a subject, and of the law mastering its absence. When the field of society extends to the point of absence, of the invisible, the inaudible, and the immemorable, when the local community is dislocated to the point where individuals no longer appear to one another, become capable of being imperceptible, the age of writing begins” (281-282). In his analysis, to thinkers like Warburton and Condillac, economic requirements of expanding information and knowledge drove the evolution of writing through pictograph, hieroglyph, abbreviated hieroglyph, alphabet to formalized algebras based on idealization, or per Baudrillard, simulacra (285-286). Consequently, the cyberspace epoch, where data storage economies are transformed by extreme inscription and high speed automatic computing, consummates a long historical movement.

A number visual practices besides literacy are linked to the development of the Western self that will eventually be referred to as the liberal humanist subject. As Ong explains, “intelligence is relentlessly reflexive, so that even the external tools that it uses to implement its workings become 'internalized', that is, part of its own reflexive process” (80). Robert D. Romanyshyn argues that perspectival viewing was originally an invention that became a deeply rooted cultural habit so that it seems natural. Thus, the Alberti-based subject led to the despotic eye of the mind, yielding the ego consciousness through Descartes' purification (349-350). Kitchin and Dodge offer the example of Charles Booth's poverty map of 1890s London demonstrating the gradual quantification of society, implying that rational subjectivity was in part produced through the objectifying panoptic gaze (82). Combining the visual and audible senses, phonetic reading, which Kittler argues was a technological invention of the 1800s, produced standard pronunciation from interiority of sound, and creates an automaton from the human reader (*Discourse Networks 1800-1900* 36-37). Jonathan Sterne follows suit with the cultivation of listening habits, which he calls ensoniment.

Foreshadowing the discussion of cybernetics in the second half of this chapter, the connection between phonetic reading, programming, and automatons should not be missed. Kittler writes of the mother tongue, "*Mama* did not indicate, as it would a century later, the existence of a children's language beyond any national language, which could contribute to a general linguistics. Instead, it was pronounced by parents only so that it might recur in children's mouths as a signature for the new education. What occurred, then, was true programming, which could thus be continued by automatons" (49). Moreso than the implicit training of perspectival viewing, phonetic reading is literally text-to-speech synthesis, like playing from a musical score. Human subjectivity transformed along with this technology to the point that it, too, seems natural. "An accomplished Mother's Mouth at the end of its self-education no longer works in an empirico-dialectical manner, but becomes the mouthpiece of an original voice sound that generates all others" (35).[18]

18 It is here that Ong, acknowledging his debt to Saussure, Parry and Havelock for calling attention to oral speech and its differences with written speech, augmented the academic discipline of textuality studies with an

This connection between the Western conception of the human individual and literacy is foregrounded repeatedly in poststructuralist and postmodern philosophy. Hayles exclaims in *How We Became Posthuman* that “one contemporary belief likely to stupefy future generations is the postmodern orthodoxy that the body is primarily, if not entirely, a linguistic and discursive construction.” (192). Many theorists implicitly agree by connecting metaphors for the mind and soul to inscription technologies. Jay David Bolter contends that “Descartes' reasoning agent can be understood as a writer who inscribes and therefore takes responsibility for his mental text,” such that “Cartesian philosophy provides a philosophical foundation for the classic age of printing, in which the author indeed both validates and is validated by the texts he publishes” (*Writing Space* 195-196). The private act of writing generates the subject at the seat of cognition who records what it sees.

Bolter, Kittler, and Derrida all single out Freud's explicit comparison of human memory to the Mystic Writing Pad, upon which marks can be semi-permanently inscribed and then erased by lifting the film, leaving faint traces on the substrate that become the object of psychoanalytic investigation. Indeed, the history of conceptions of the human psyche seems connected to the history of modes of inscription. Kittler frequently points out that Nietzsche noted that our writing tools are also working on our thoughts, and was something of a cybersage prototype, composing with an early model typewriter invented by Malling Hansen and, after it broke down, philosophizing about it. “And this philosophy,” Kittler writes in *Gramophone, Film, Typewriter*,

awareness of technological media, made explicit by McLuhan. The connection between text-to-speech synthesis by electronic machinery and human phonetic reading will be further explored in the context of the critical programming study of the ensoniment of Plato's *Symposium* in chapter four.

“instead of deriving the evolution of the human being from Hegel's spirit (in between the lines of books) or Marx's labor (in between the differential potential of muscular energy), began with an information machine” (208).

Writing tools alone do not make the modernist subject. Foucault, to whom Hayles gives credit through his detailed historical studies, demonstrated how the encoding of logical and semiotic structures into living organisms requires enormous apparatus of social control. Similarly, to Hardt and Negri the multitude is transformed into orderly totality by administrative bureaucracy machines; the state produces society through biopower (87-89). This is a view of the soul as the contingent correlative to technologies of power over body, the effect and instrument of political anatomy, entrained in the prison of the body (*Discipline and Punish* 29-30). Amending Hayles' discursive reductionism, it is worth recalling how Foucault appeals to the broad spectrum concept of biopower to explain how the modern state produces citizens and subjectivity. “The individual is no doubt the fictitious atom of an 'ideological' representation of society; but he is also a reality fabricated by this specific technology of power that I have called 'discipline'. We must cease once and for all to describe the effects of power in negative terms: it 'excludes', it 'represses', it 'censors', it 'abstracts', it 'masks', it 'conceals'. In fact, power produces; it produces reality; it produces domains of objects and rituals of truth” (*Discipline and Punish* 194).

Disciplinary society transforms human beings, producing docile bodies and minds. Foucault urges this narrative of meticulous subordination, permanent coercions, progressive training, and automatic docility be added to familiar histories of ideas based on the state of nature, social contract, and general will. Man the machine develops through the anatomico-metaphysical registers of philosophers, and disciplines of technico-political registers, bodily improving improvement, creating the man of modern humanism, yielding "a body is docile that may be subjected, used, transformed and improved. The celebrated automata, on the other hand, were not only a way of illustrating an organism, they were also political puppets, small-scale models of power: Frederick II, the meticulous king of small machines, well-trained regiments and long exercises, was obsessed with them" (*Discipline and Punish* 136). A carceral archipelago and subtle, graduated carceral net are good images for disciplinary society. Described as a micro-physics of power forming a political technology of body, with control operations likened to those of machines, Foucault concludes that “from such trifles, no doubt, the man of modern humanism was born” (*Discipline and Punish* 141).

The postmodern critique of subjectivity questions the linkages between textuality, and semiotics in general, to the soul, as well as the stability, homogeneity, and inevitability of the social arrangements of disciplinary bureaucracies that Foucault's studies illuminated, on to liberal democratic capitalism. Jameson contends that “the sentence sequence leaves the reading mind without an object, which it therefore conveniently supplies itself in the form of an ideal or imaginary literary referent, a kind of subliminal or archetypal image in which a colorless surface oscillates back and forth in time between dull indistinction and the heightened perception of varied points. . . . In effect, the reader seems unable to conclude that language has broken down (something which would leave her or him without any subject position whatsoever), and therefore as in a reverse shot in film constructs some new imaginary object to justify the persistence of the subject position already achieved” (136). Thus Foucault employed the *Las*

Meninas painting to illustrate the *virtuality* of the subject position, an epiphenomenon of the likewise virtual object position in which it is situated, “not some mere perceptual aggregate of physical things, but a social configuration or ensemble of social relationships (even physical perception and seemingly rock-bottom experiences of the body and of matter being mediated by the social). What one concludes from such an argument is not that the unified subject is unreal or undesirable and inauthentic, but rather that it is dependent for its construction and existence on a certain kind of society and is menaced, undermined, problematized, or fragmented by other social arrangements” (137).

In Jameson's analysis, reading separates into distributed operations, including those dealing with the material signifiers themselves, which are objects imbued with histories, such that philosophical study shifts from problems of linguistics to the image society and the media, such that “it would seem to consist essentially in the inevitable presence of noise as such within any communicational system“ (141).[19] He reiterates Lyotard, who described this diminishment of the autonomous, liberal subject in *The Postmodern Condition*. “A self does not amount to much, but no self is an island; each exists in a fabric of relations that is now more complex and mobile then ever before. Young or old, man or woman, rich or poor, a person is always located at nodal points or specific communication circuits, however tiny these may be. Or better: one is always located at a post through which various kinds of messages pass” (15). This nuanced interpretation, however, is overshadowed by Jameson's primary conception of postmodernism as

19 By invoking concepts from cybernetics, Jameson works at the periphery of what I call the philosophy of computing. It is telling that he quotes William Gibson in the introduction to *Postmodernism*, and in a footnote laments that there was no time to write a chapter on cyberpunk.

it relates to capitalism, as "the consumption of sheer commodification as a process. The life-style of the superstate therefore stands in relationship to Marx's fetishism of commodities as the most advanced monotheisms to primitive animisms or the most rudimentary idol worship" (x).

Awareness of a regressive tendency in Western subjectivity predates these postmodern articulations. Often equivocated with Nietzsche's last man and explained in terms of Marxist commodity fetishism, blinking its way through life grasping at the comfortable crutches provided by religious, cultural, social, and especially consumer rewards, it is the much maligned outcome of Horkheimer and Adorno's dialectic of Enlightenment, having been transformed along with the work of art in the age of mechanical reproduction. Walter Benjamin grounds much of critical media theory by analyzing the withering aura of the work of art picked up by many postmodern theorists, especially Baudrillard, who declares "no more medium, no more image any more than an industrial object is the mirror the identical one that succeeds it in the series" (*Simulacra and Simulation* 97). To Benjamin theater, transformed into cinema, loses its dynamic interaction with the audience and art takes on more of a bureaucratic, industrial production process, following the logic of Marxist alienation of labor. Group reception including feedback is not possible with paintings and other individual pieces not easily reproducible or mass communicated.

With mechanical reproduction we thus produce mass entertainment like cinema, radio, television, print rather than individual spectacles that cannot be recorded, and spectacles become mass spectacles like war and later rock concerts. "Quantity has been transmuted into quality. The greatly increased mass of participants has produced a change in the mode of participation. . . . Clearly, this is at bottom the same ancient lament that the masses seek distraction whereas art demands concentration from the spectator” (XV). Participation shifts to passive consumption, reception in a state of distraction. Moreover, Benjamin discovers the extreme closeup and other cinematic techniques are ways perceptions change through technology moreso than though society, foreshadowing the concept of synaptogenesis that will be key to later posthuman theorists like Hayles.

However, changes in habitual perception urged by the social transformations accompanying industrialization and commodification remain Benjamin's focus, along with that of Horkheimer and Adorno. To them, the core of the symbolic is nature as self-representation, but this original representation has been replaced by universal fungibility. "Enlightenment stands in the same relationship to things as the dictator to human beings. He knows them to the extent that he can manipulate them. The man of science knows things to the extent that he can make them. . . . An atom is smashed not as a representation but as a specimen of matter, and the rabbit suffering the torment of the laboratory is seen not as a representative but, mistakenly, as a mere exemplar" (6-7). This leveling rule of abstraction creates a herd; “earlier, fetishes had been subject to the law of equivalence. Now equivalence itself becomes a fetish” (12). The means of production become fetishized as well, and the festival is reduced to farce, the feast to a government sanctioned holiday. “Magic passes into mere activity, into the means in short, into industry. The formalization of reason is merely the intellectual expression of mechanized production. . . . The general overflowing is no longer possible. The period of turbulence has been individualized. Holidays have supplanted the feast. In fascism they are supplemented by the collective fake intoxication, concocted from radio, headlines, and Benzedrine" (81-83).

Stepping through their dialectic, mathematics becomes reified thought as a machine process, and all phenomena reflect the imprint of a technologized schema, so that the engineers and scientists who control the jargon, along with world rulers, see themselves as architects of world history. “The true masters, as both producers and reproducers, are those who speak the jargon with the same free-and-easy relish as if it were the language it has long since silenced. Such is the industry's ideal of naturalness. . . . The general designation culture already contains, virtually, the process of identifying, cataloging, and classifying which imports culture into the realm of administration” (101-104). This nominalism can be seen as prototypical bourgeois thinking, which Foucault later refers to as the order of things, and others, as formalism and functionalism.

In developing this argument Horkheimer and Adorno invoke the image of the mythological hero Odysseus. The direct power of words over things is broken in the myth of Odysseus and the cyclops, in which the former escapes the latter by declaring his name to be Nobody. At sea, however, the nominalism of his royal lineage leaves him defenseless; the gods ignore his pleas. “Odysseus's defenselessness against the foaming sea sounds like a legitimation of the enrichment of the voyager at the expense of indigenous inhabitants. Bourgeois economics later enshrined this principle in the concept of risk: the possibility of foundering is seen as a moral justification for profit” (48). Finally, when he has his crew fill their ears with wax so they can pass by the Sirens, they become a metaphor for the trapped office workers of bourgeoisie society. “Just as he cannot give way to the lure of self-abandonment, as owner he also forfeits participation in work and finally even control over it, while his companions, despite their closeness to things, cannot enjoy their work because it is performed under compulsion, in despair, with their senses forcibly stopped. . . . Humanity, whose skills and knowledge become differentiated with the division of labor, is thereby forced back to more primitive anthropological stages, since, with the technical facilitation of existence, the continuance of domination demands the fixation of instincts by greater repression” (27-28).

Taking arguments made by Freud, Marcuse, and others who reach the dialectical paradox of civilization and its discontents, Horkheimer and Adorno see progress itself reverting to regression by elevating the metaphysical status of commodities. “If, in the absence of the social subject, the volume of goods took the form of so-called overproduction in domestic economic crises in the preceding period, today, thanks to the enthronement of powerful groups as that social subject, it is producing the international threat of fascism: progress is reverting to regression. That the hygienic factory and everything pertaining to it, Volkswagen and the sports palace, are obtusely liquidating metaphysics does not matter in itself, but that these things are themselves becoming metaphysics, and ideological curtain, within the social whole, behind which real doom is gathering, does matter" (xviii). With technical capabilities directed to bloated entertainment apparatus rather than abolishing hunger, amusement becomes an ideal, comparable to Benjamin's conclusion that cyclical war is needed to satisfy the productive engines of society that has abandoned its higher purposes. Yet the subjects trapped in bourgeois comforts hear muted Siren songs of commercialized art from which the consumer is alienated and montage reigns. Culture has become advertising. "Through the language they speak, the customers make their own contribution to culture as advertising. For the more completely language coincides with communication, the more words change from substantial carriers of meaning to signs devoid of qualities; the more purely and transparently they communicate what they designate, the more impenetrable they become. The demythologizing of language, as an element of the total process of enlightenment, reverts to magic" (133-136).

Adorno continues this tale of regressive subjectivity in his essay “On the Fetish-Character in Music and the Regression of Listening.” Artistic work is reduced to a signature melody that can be reified as intellectual property, and a musical children's language suitable for surface enjoyment. The listener is converted into acquiescent purchaser, whose experience is ultimately shaped by fetishized monetary capital as the cost of listening. In the regression of listening, *deconcentration* foreshadows the distracted attention characteristic of our current obsession with mobile technologies. "They fluctuate between comprehensive forgetting and sudden dives into recognition. They listen atomistically and dissociate what they hear, but precisely in this dissociation they develop certain capacities which accord less with the concepts of traditional aesthetics than with those of football and motoring" (286-287).

These new listeners have free time and no freedom, like housewives surrounded by state of the art, labor saving yet time consuming appliances. Regressive subjectivity of the former Odysseus degrades into this jack of all trades, master of none, the *bricoleur* beloved by poststructuralists, but with a pejorative connotation. “Regressive listeners have key points in common with the man who must kill time because he has nothing else on which to vent his aggression, and with the casual laborer. To make oneself a jazz expert or hang over the radio all day, one must have much free time and little freedom. . . . The new listeners resemble the mechanics who are simultaneously specialized and capable of applying their special skills to unexpected places outside their skilled trades” (294). The individual is liquidated between incomprehensibility and inescapability, and consciousness is defined by displeasure in pleasure.

Two thousand years ago Diogenes Laertius, in his *Lives of Eminent Philosophers*, relates an account of Pythagoras where “Sosicrates in his *Successions of Philosophers* says that, when Leon the tyrant of Philius asked him who he was, he said, 'A philosopher', and that he compared life to the Great Games, where some went to compete for the prize and others went with wares to sell, but the best as spectators; for similarly, in life, some grow up with servile natures, greedy for fame and gain, but the philosopher seeks for truth” (8). While the contemplative observer who philosophizes certainly exists out there among the mass of spectators, the majority are there for amusement and consumption. Moreover, broadcast media further separates the audience from the spectacle itself, conveniently segregating the masses in their domiciles, eateries, and stadiums like Odysseus tied to the mast of his ship.

Mixing these themes together, Adorno credits a new fetish in the technical production of perfect performance, leading to personal worship of home theaters as emblematic of regressive subjectivity's retreat from the heart of the spectacle to its peripheries. He plies his harshest, blistering criticism of fetishistic listeners to radio hams, which is worth quoting at length on account of their foreshadowing computer enthusiasts. “As radio ham he becomes the discoverer of just alone those industrial products which are interested in being discovered by him. . . . Of all fetishistic listeners, the radio ham is perhaps the most complete. It is irrelevant to him what he hears or even how he hears; he is only interested in the fact that he hears and succeeds in inserting himself, with his private equipment, into the public mechanism, without exerting even the slightest influence on it. . . . He pictures himself as the individualist who whistles at the world. But what he whistles is its melody, and his tricks are less inventions of the moment than stored-up experiences from acquaintance with sought-after technical things" (292-293). The similarity to Bauerlein's criticism of youth detailed in the previous chapter, self-absorbed in social media is unmistakable, as well as allusions to the postmodern precession of simulacra that Chun now calls programmed visions, which will be examined in the next chapter. Baudrillard describes the situation as “an about-face through which it becomes impossible to locate one instance of the model, of power, of the gaze, of the medium itself, because *you* are always already on the other side. No more subject, no more focal point, no more center or periphery: pure flexion or circular inflexion” (*Simulacra and Simulation* 29).

We are ahead of ourselves. The situation as cast by Horkheimer and Adorno at the onset of postmodernity was more grim than the consumer horde ridiculed by later theorists. Remember I began with the Dehomag advertisement from Nazi Germany. They wrote, “the individual had become an impediment to production. The lack of synchronicity between technical and human development, the 'cultural lag' which used to exercise the minds of sociologists, is beginning to disappear. Economic rationality, the vaunted principle of the smallest necessary means, is unremittingly reshaping the last units of the economy: businesses and human beings. The most advanced form at a given time becomes the predominant one. . . . In the progress of industrial society, which is supposed to have conjured away the law of increasing misery it had itself brought into being, the concept which justified the whole – the human being as person, as the bearer of reason – is going under. The dialectic of enlightenment is culminating objectivity in madness” (167-169).

Horkheimer and Adorno propose that *ticket thinking*, which emerged in industry and business to better synchronize technical and human development, has been extended to international relations, referring to Germany's struggle for raw materials to modernize and militarize following the first Word War, and ultimately to the holocaust itself in an effort to eliminate the Jewish middlemen of the prior economic era. Tickets in this passage are small cards to represent and also summarize party allegiance, like the tickets used to board railroads, enter theaters and sports arenas. Automated ticket thinking, epitomized by IBM punch card machinery, becomes the prototype of the postmodern technological era in which all interests—personal, corporate, institutional, national—are engineered to fit onto the tickets. “The Jewish middleman fully becomes the image of the devil only when economically he has ceased to exist. Victory is thus made easy, and the anti-Semitic family man becomes the spectator, exempt from responsibility, of an irresistible historical tendency, intervening only when called to do so by his role as an employee of the Party or the Zyklon gas factories” (171).

Horkheimer and Adorno lived through violent times, and witnessed first hand the atrocities a technologically enhanced, industrial scale military could commit. Although Lyotard wrote his report at the behest the of Canadian government, the United States is often the locus of postmodern critique, especially by Continental philosophers. Lyotard, in detailing the defining document of the postmodern condition, modulated his cautionary findings with some hopeful comments, and his subsequent writings reflect an ambivalence to a period that others have declared abysmal, the end of times. A visual illusion of a duck/rabbit is often used to illustrate this dual position. The duck/rabbit of postmodernism, sensed in the powerlessness and dispersion of the subject, the desire to seize reality and return to terror, has a reverse face that is waging war on totality, witnessing the unpresentable, and saving the name. “The emphasis can be placed on the powerlessness of the faculty of presentation, on the nostalgia for presence felt by the human subject, on the obscure and futile will which inhabits him in spite of everything. . . . The emphasis can also be placed on the increase of being and the jubilation which result from the invention of new rules of the game, be it pictorial, artistic, or any other. . . . Here, then, lies the difference: modern aesthetics is an aesthetic of the sublime, though a nostalgic one" (*Postmodern Condition* 79-80).

Indeed, Adorno's radio ham is a caricature of the maker spirit of the free, open source epoch. Histories of the personal computer, operating systems, software applications, networking, and the Internet itself hearken back to the amateur radio operators, model train enthusiasts, and electronic tinkers, through whose practiced obsessions new rules, new games, and new modes of engaging technology arose, ushering in the *post*-postmodern. Referring to these descendants of the early radio hams, in 2000 Freiberger and Swaine conclude *Fire in the Valley: The Making of the Personal Computer*, “in one way or another, they were all dreaming of one thing: the personal computer, the packaging of the awesome power of computer technology into a little box that anyone could own. Today, changing the world is the little machine's job. The personal computer, once a truly revolutionary idea, has become a commonplace tool” (424). Untouched by the negativity of postmodern critique, they continue: “just as the World Wide Web was invented on a NeXT cube, it is likely that the next technological revolution will be invented on a personal computer. Probably by some bright young hacker. She may even be reading this book right now” (424). The next step in this progression through regressive subjectivity towards the cyborg dividual is to explain why the United States became the primary location for its development and consummation.

Inventing the Posthuman in Heidegger's America

Martin Heidegger maintains a tenuous position in the philosophical pantheon. Despised for his cozy alignment with the Nazi party after being granted a top university position, his writings are nonetheless referenced in nearly every philosophy of technology anthology for giving the discipline its metaphysical foundation. He also lived and theorized in the liminal boundary of predigital culture, when prognostications about the computerized future overran what was empirically attainable, yet he remained a stalwart adherent to a more pure, mythical past of forest paths and handiwork, especially when it came to seeking places to frame and conduct philosophical investigations. He gave the philosophy of technology the concepts of *enframing* and *standing reserve* to articulate the ontological position reached by industrial societies: “the machines and apparatus are no more cases and kinds of Enframing than are the man at the switchboard and the engineer in the drafting room. Each of these in its own way indeed belongs as stockpart, available resource, or executor, within Enframing; Enframing is never the essence of technology in the sense of a genus. Enframing is a way of revealing having the character of destining” (“The Question Concerning Technology” 29). His often cited depiction of a silver chalice in “The Question Concerning Technology” serves to frame his study of the essence of machine technology, but his Nietzsche lectures provide a starker backdrop for framing the posthuman.

It is noted by the translator of this four volume series of lectures that “Nietzsche's caustic reduction of Being to a vapor and a fallacy appears to have provoked, almost singlehandedly, Heidegger's ensuing lecture courses,” which followed the publication of *An Introduction to Metaphysics* (*Volume 4* 182). From these studies of Nietzsche's unpublished fragments that became *The Will to Power*, Heidegger mused that we humans ourselves belong to the command of nihilism, included in the psychological reckoning and calculation left in the wake of prior metaphysics and the evaporation of Being, despite his ethical conclusion that man is never merely standing-reserve. “He is without issue because he is always thrown back on the paths that he himself has laid out: he becomes mired in his paths, caught in the beaten track, and thus caught he compasses the circle of his world, entangles himself in appearance, and so excludes himself from being" (130).

If Nietzsche's parody of the Last Man in *Thus Spoke Zarathustra* embodies regressive subjectivity, his vaunted Overman embodies the paradoxical combination of destructive nihilism and technical, creative savvy that flourished in the late 1990s as cyberpunk. “What is needed is a form of mankind that is from top to bottom equal to the unique fundamental essence of modern technology and its metaphysical truth; that is to say, that it lets itself be entirely dominated by the essence of technology precisely in order to steer and deploy individual technological processes and possibilities” (117). There are no more metaphysics; all thinking is conscious, logical, and optimizing. “What Nietzsche already knew metaphysically now becomes clear: that in its absolute form the modern machine economy, the machine-based reckoning of all activity and planning, demands a new kind of man who surpasses man as he has been hitherto” (116). A Heideggerian posthuman is a fantasy of what will inevitably happen by default from the triumph of Nietzschean metaphysics. “Western logic finally becomes logistics, whose irresistible development has meanwhile brought forth the electronic brain, whereby man's nature and essence is adapted and fitted into the barely noticed Being of beings that appears in the nature of technology" (*What is Called Thinking?* 238). Heidegger foregrounds the invisibility of technology and its systemic rootedness.

In making this seemingly obligatory passage through Heidegger I am following many others, including Verena Andermatt Conley, who in the preface to the anthology *Rethinking Technologies* argues we must go beyond thinking “in terms of domination of nature and of loss of humanness by way of technology, but neither of transformation of subjectivities nor of limits imposed by natural or social ecology. . . . At stake is the singularity of the subject threatened with annihilation as much through technology as through the effects of mass media. The individual has a false feeling of autonomy and agency, while he or she is being drugged or manipulated into immobility” (xiii).

Don Ihde presents a more detailed progression of his brief history of the philosophy of technology, but agrees that its first wave, which included Heidegger, Foucault, Habermas, Ellul, and Marcuse, foregrounded threats of autonomous technology, subsumption of all other styles into calculative thinking, docile social control, and technocratic consciousness, conjoined with late capitalist and industrial socialist ideologies exemplified by the closed world discourses of the Cold War (33). Nonetheless, Iain Thomson calls for one last look at Heidegger, in particular his critique of America, where the danger lurks and perhaps the saving power resides. His analysis is tied up in Heidegger's “profound but idiosyncratic conception of metaphysics as ontotheology. . . . He builds upon the Kantian idea that we implicitly participate in the making-intelligible of our worlds, but maintains that our sense of reality is mediated by lenses we inherit from metaphysics. . . . These ontotheologies provide the dual anchors that suspend humanity's changing sense of 'reality', holding back the flood waters of historicity long enough to allow the formation of an 'epoch', a historical constellation of intelligibility which is unified around its ontotheological understanding of the being of entities” (150).

Heidegger advances a doctrine of ontological holism privileging the human perspective, albeit one that in the technological epoch relates to the transformation of beings into intrinsically meaningless resources for digitization and object-oriented design. This simplistic view leads to a parallel, shallow cyborg stereotype based on cosmetic, psychophamarcological, cybernetic enhancement. Thomson contends that the danger is continuous improvement as a totalizing philosophy, the problem of the 'happy enframer' epitomized with America of the late 1960s. “By 1969, however, at the height of the Vietnam War, there no longer seems to be any question in Heidegger's mind: 'America' has become virtually synonymous with 'the danger'" (154). Backing up his claim is Hubert Dreyfus, who argues that the inability to control the very drive to control expresses the definitive ontotheology of our age, like the drive to build artificial intelligence (154).

Crucially, this is not a Luddite position: the greatest danger is that we get a symptom-free disease treated by the very thing that makes us ill, which Zizek illustrates with the figure of a chocolate-flavored laxative. I hope by now you have a sense of coming full circle to the position Bauerlein reached concerning the dumbest generation with which I began. Is this not a view of the present situation Kittler seeks? Ontotheologies undermining meaningfulness of our sense of reality, with symptoms like environmental devastation, obsession with biogenetic perfection, empty optimization imperatives. It remains to name our post-postmodern condition, and to do so we must detach from the ontological holism that puts the human biological entity at the center of all meaning and embrace an extended mind.

Hayles contends that we in America have already become posthuman, from developments in cybernetics coinciding with the period during which Heidegger flourished. She provides a four point summary of the posthuman view that is worth quoting at length, for it sets up the articulation of the post-postmodern network dividual cyborg to grasp the as-is situation and propose solutions to the regressive track Bauerlein and many others fear we are following. “First, the posthuman view privileges informational pattern over material instantiation, so that embodiment in a biological substrate is seen as an accident of history rather than an inevitability of life. Second, the posthuman view considers consciousness regarded as the seat of human identity in the Western tradition long before Descartes thought he was a mind thinking, as an epiphenomenon, as an evolutionary upstart trying to claim that it is the whole show when in actuality it is only a minor sideshow. Third, the posthuman view thinks of the body as the original prosthesis we all learn to manipulate, so that extending or replacing the body with other prostheses becomes a continuation of a process that began before we were born. Fourth, and most important, by these and other means, the posthuman view configures human being so that it can be seamlessly articulated with intelligent machines” (*How We Became Posthuman* 2-3). The remainder of this chapter will address the first two points, and the theoretical framework and methodology developed in the next chapter will seek to flip the duck/rabbit view of regressive subjectivity to realign the human-computer symbiosis into a mutually advantageous trajectory.

Network Dividual Cyborg

The regressive subjectivity leading to the dumbest generation reflects how we in Heidegger's America became posthuman. The story is not really told through the musings of the little German humanist squirreled away in his mountain hut, however, but through those in tune with its boisterous, market-driven industry and commerce.[20] Marshall McLuhan leads the advance of a new breed of theory stylistically adapted to its subject matter, a combination of short snippets of philosophy, science, social theory, entertaining prose, images, and even some noise. He continues the advance where Horkheimer and Adorno see the dialectic of enlightenment regressively turning back on itself by contrasting unifying automation to fragmenting mechanization, and electrical illumination to the candle lit enlightenment of modernity. His famous slogan that media are the extension of man develops from his training in classical scholarship and textuality studies conjoined with his understanding that the central nervous system is radically externalized, making our lives information processes. “Our private and corporate lives have become information processes because we have put our central nervous systems outside us in electric technology. . . . Our central nervous system is not merely an electric network, but it constitutes a single unified field of experience” (60; 302). McLuhan recognizes societies of control supplanting their disciplinary foundations, where illumination is

20 In a recent article, Christian Fuchs argues that the damning evidence of Heidegger's systematic anti-Semitism revealed by the newly published series of “Black Notebooks” obligates philosophers of technology and media theorists to deprecate and henceforth avoid his work or mention of his name.

productive like Foucault's biopower. “Yet this organic unity of interprocess that electromagnetism inspires in the most diverse and specialized areas and organs of action is quite the opposite of organization in a mechanized society. . . . Energy and production now tend to fuse with information and learning. Marketing and consumption tend to become one with learning, enlightenment, and the intake of information. . . . The electronic age is literally one of illumination“ (302-304).

Automation invades the mechanical world by the instantaneous character of electricity, and with computers, autonomous execution in the physical world based on dynamically reprogrammable simulations, described by McLuhan as feedback servo-mechanist structures. His early vision of artificial intelligence, however, maintains the contours of the human consciousness it reflects. “Any process that approaches instant interrelation of a total field tends to raise itself to the level of conscious awareness, so that computers seem to think. . . . But a conscious computer would still be one that was an extension of our consciousness, as a telescope is an extension of our eyes, or as a ventriloquist's dummy is an extension of the ventriloquist” (305). Thus, McLuhan makes the interesting suggestion that increasing complexity and abstraction of programmed machinery leads to more general capabilities, like the hand versus the paws, but retains striated trajectories extending human purposes. Automation of the industrial plant is a model of what will happen in society at large (311). Thus his exemplar for programmability is the comical Schmoo fantasy realized by the automated movement of information. “These traits of store, or memory, and accelerator are the basic features of any medium of communication whatever. In the case of electricity, it is not corporeal substance that is stored or moved, but perception and information. . . . We have now only to name and program a process or a product in order for it to be accomplished. Is it not rather like the case of Al Capp's Schmoos? One had only to look at a Schmoo and thinking longingly of port chops or caviar, and the Schmoo ecstatically transformed itself into the object of desire. Automation brings us into the world of the Schmoo" (305).

McLuhan's views tie in neatly with Baudrillard. Custom-built supplants mass-produced, but the product remains a concretization of large scale industrial and commercial forces, leaving the human in a constrained subject position. This position is comparable to Benjamin's loss of aura through mechanized reproduction and the phenomena Kittler resignedly concludes are at the heart of media convergence. When we talk about cybernetics we imply networks as that opposite to humans, really the built environment surrounding human bodies, but will find that embodiment has been influenced by computing as much as computing has been shaped by not only the immateriality of Cartesian consciousness, but also the spatiotemporal characteristics of bodies themselves. There are no better reviews of this development than the contested narratives of the origins of cybernetics in the United States told by N. Katherine Hayles, and the accompanying closed world politics of the Cold War told by Paul N. Edwards, which together will form the focus of the next section.

Closed World Cybernetics

While the as-is condition today relates to *cyberspace*, we are reminded by Manovich that this made-up word derives from the prior neologism *cybernetics* from Norbert Weiner, who in his 1947 book bearing that title “defined it as the science of control and communication in the animal and machine. . . . He derived the term *cybernetics* from the ancient Greek word *kybernetikos*, which refers to the art of the steersman and can be translated as good at steering" (*Language of New Media* 251). Carl Mitcham adds that "cybernetics, as the theory of the way information states interact with one another to produce certain behaviors, explains the nature of the technological in terms of information processing and proposes to provide a means to guide or direct this processing" (206). At the heart of cybernetics is the idea of the closed-loop feedback circuit, in which instruments are connected to the output of a control operation to further regulate the mechanism until the desired outcome is attained. Alexander Galloway claims that "Foucault introduced the concept of biopower to help explain this phenomenon. His formulation was consistent with the functioning of protocol, for biopower is the power to interpret material objects as information, to affect objects at the statistical or informational level, not at the level of individual content" (69). Lyotard, too, invokes cybernetics as a key constituent in the development of functionalism later applied to social theory: "it took yet another turn in the 1950s with Parsons's conception of society as a self-regulating system. The theoretical and even material model is no longer the living organism; it is provided by cybernetics, which, during and after the Second War War, expanded the model's applications. . . . The true goal of the system, the reason it programs itself like a computer, is the optimization of the global relationship between input and output in other words, performativity. . . . The only alternative to this kind of performance improvement is entropy, or decline" (*Postmodern Condition* 11-12).

A key difference between traditional philosophical logic and cybernetics is that the latter involves engineered dynamism, logic *plus* conditional control. Before such prototypes of modern digital computers provided concrete manifestations of cybernetic principles, electromechanical relay devices for telephone switching circuits and differential equations networks analysis were theorized, built, and used by Hazen, Bush, and Shannon (Misa 155). According to Kittler, "computer algorithms, instead of simply reproducing a logic, consist of LOGIC + CONTROL. . . . Obviously, [Turing's] COLOSSUS beat binary addition with binary addition, but even the first computer in the history of science or warfare would have been nothing but a several-ton version of Remington's special typewriter with a calculating machine had it not observed conditional jump instructions. . . . Conditional jumps, first envisioned in Babbage's unfinished Analytical Engine of 1835, were born into the world of machines in 1938 in Konrad Zuse's apartment in Berlin, and this world has since been self-identical with the symbolic" (*Gramophone Film Typewriter* 248).

A second, more subtle shift involves the equivocation of humans and machines under putatively similar functionalist rubrics. By the time Wiener popularized the term in the late 1940s, this new mode of perceiving the relationship between humans and the environment as engineering problems was firmly entrenched. Hayles credits the retrospectively named Macy Conferences on Cybernetics, held in New York at the Josiah Macy, Jr. Foundation from 1943 to 1954, for creating the new paradigm on which our posthuman identity is based. "To succeed, they needed a theory of information (Shannon's bailiwick), a model of neural functioning that showed how neurons worked as information-processing systems (McCulloch's lifework), computers that processed binary code and that could conceivably reproduce themselves, thus reinforcing the analogy with biological systems (von Neumann's specialty), and a visionary who could articulate the larger implications of the cybernetic paradigm and make clear its cosmic significance (Wiener's contribution). The result of this breathtaking enterprise was nothing less than a new way of looking at human beings. Henceforth, humans were to be seen primarily as information processing entities who are essentially similar to intelligent machines" (*How We Became Posthuman* 7). To Hayles, the machine-organism equation was formulated by the conference participants in ways to make the analogy work, rather than the other way around.

Alan Turing, working on the other side of the Atlantic ocean, was also complicit in this perspective. His famous Imitation Game, and the predictions in "Computing Machinery and Intelligence," launched countless projects in artificial intelligence research. He proposed "the original question, Can machines think? I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted" (55). Turing reasoned that advances in hardware technology would provide computing machinery with enough memory and speed to simulate the human brain, and that the hard problems would devolve to matters of programming. Moreover, he recommended a learning model based on a familiar view of the soul as a blank notebook to be filled through experience. "Instead of trying to programme to simulate the adult mind, why not rather try to produce one which simulates the child? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child brain is something like a notebook as one buys it from the stationers. Rather little mechanism, and lots of blank sheets" (62). He argues that two way communication is at the core of intelligence, not embodiment, noting that Helen Keller was able to communicate with her teachers; "the imperatives that can be obeyed by a machine that has no limbs are bound to be of a rather intellectual character, as in the example (doing homework) given above. Important amongst such imperatives will be ones which regulate the order in which the rules of the logical system concerned are to be applied" (63).[21]

For decades we have spoken about our brains in terms of hardware and software. To David Golumbia, this is the signature bias of what he calls the cultural logic of computation. "By turning these methods and their concomitant linguistic analyses to the terms of philosophical theory, these thinkers could mount a new assault on the traditional mind/body problem armed with a clear and yet paradoxical tool out of computer science. The brain was not exactly a computer and the mind was not exactly software, but the hardware/software model is still the best metaphor we have for describing the brain, so much so that thinking and computation are almost indistinguishable" (56-57).

Cybernetic information theory has also been accused by Lyotard of neglecting the agonistic aspect of society (*Postmodern Condition* 16), although its own history is rife with contestation, as Hayles shows through her detailed study of multiple narratives of the Macy Conferences, in particular through her fortuitous encounter with memoirs of periphery female participants. Her goal in this effort comprising a large portion of *How We Became Posthuman* is to replace the prevailing technological determinist metanarrative of cyborg identity with historically contingent stories that occurred during the development of cybernetics. "As we have seen, one way to construct virtuality is the way that Moravec and Minsky do as a metanarrative about the transformation of the human into a disembodied posthuman. I think we should be

21 The objection can be made that Turing privileges vision and hearing as exemplars of embodiment tied to intelligence, whereas other sensory means ought to be credited as well, as Clark and Hansen argue in their theories of radical embodiment.

skeptical about this metanarrative. . . . By turning the technological determinism of bodiless information, the cyborg, and the posthuman into narratives about the negotiations that took place between particular people at particular times and places, I hope to replace a teleology of disembodiment with historically contingent stories about contests between competing factions, contests whose outcomes were far from obvious" (22). For example, the first wave of cybernetics privileged homeostatis over reflexivity, "largely because it was more manageable quantitatively. Reflexivity lost because specifying and delimiting context quickly ballooned into an unmanageable project. . . . If humans are information-processing machines, then they must have biological equipment enabling them to process binary code. The model constructing the human in these terms was the McCulloch-Pitts neuron" (56-57).

Humans and machines became increasingly linked as endocrine models lost favor to electronic theories of complex, high speed control systems that were deemed superior on account of the reasonable analog of the McCulloch-Pitts neuron. "Thus, a simplification necessitated by engineering considerations becomes an ideology in which a reified concept of information is treated as if it were fully commensurate with the complexities of human thought" (54). As the human got constructed in terms of the machine, man became the portable instrument set, solving the problem of how to test cybernetic theories with the crude, expensive electronic machinery available. To Hayles, it was a shift from a laboratory white box to a human black box, now understood as an analogical white box to which cybernetic adjustments can be applied. Indeed, in later years Wiener recognized that cybernetics was eroding the liberal subject as locus of control. "Concluding that we are too much in tune with the objects of our investigations to be good probes, Wiener counsels that cybernetics had best be left to the physical sciences, for to carry it into the human sciences would only build exaggerated expectations. . . . By the 1960s, the link between liberal humanism and self-regulation, a link forged in the eighteenth century, was already stretched thin; by the 1980s, it was largely broken” (110-112).

Tradeoffs occurred in the second wave of cybernetics as well. Its key theorists, according to Hayles, were Humberto Maturana and Francisco Varela. They sought to demonstrate what philosophers had sensed for a long time, that what matters phenomenologically is the point of view of feedback control system input, not some objective reality. Under this perspective “each living system thus constructs its environment through the domain of interactions made possible by its autopoietic organization. What lies outside the domain does not exist for that system” (137). At the same time, Maturana defined 'living' in ways that afford machines to be considered alive as autopoietic physical systems, something which von Neumann had long mused as well. “To account for a system's embeddedness in an environment, Maturana uses the concept of structural coupling. . . . Thus, time and causality are not intrinsic to the processes themselves but are concepts inferred by an observer. . . . Information, coding, and teleology are likewise inferences drawn by an observer rather than qualities intrinsic to autopoietic processes. . . . One implication of letting go of causality is that systems always behave as they should, which is to say, they always operate in accord with their structures, whatever those may be” (138-139).

Autopoietic theory alters our notions of autonomy and reflexivity. An observer structurally coupled to phenomena is a strike against the detached, ocularcentric, liberal humanist subject. “Because Maturana understands self-consciousness solely in linguistic terms, seeing it as an emergent phenomenon that arises from autopoietic processes when they recursively interact with themselves, consciousness for him becomes a epiphenomenon rather than a defining characteristic of the human as an autopoietic entity. The activity of cerebration represents only a fraction of the total autopoietic processes, and self-consciousness represents only a fraction of cerebration. Thus the theory implicitly assigns to consciousness a much more peripheral role than it does to autonomy and individualism. In this respect, autopoietic theory points toward the posthuman even as it reinscribes the autonomy and individuality of the liberal subject” (145). Hayles concludes that closure and reflexivity supplant self-possession as grounding the subject's identity the way it had for the liberal subject within industrial capitalism: “information's body is still contested, the empire of the cyborg is still expanding, and the liberal subject, although more than even an autonomous individual, is literally losing its mind as the seat of identity" (149).

Hayles makes a bravo performance in eloquence and philosophical depth concerning this transition to a probabilistic worldview active in information theory throughout *How We Became Posthuman*, and concerning the danger implicit in its infiltration into social and cognitive science. “Statistical and quantum mechanics deal with uncertainty on the microscale; communication reflects and embodies it on the macroscale. *Envisioning relations on the macroscale as acts of communication was thus tantamount to extending the reach of probability into the social world of agents and actors*” (90). The danger is informational ideology; “the computational universe becomes dangerous when it goes from being a useful heuristic to an ideology that privileges information over everything else" (244).

Many other critical theorists echo her point. Ihde, the prominent philosopher of technology, argues that “the use of the computer, while not determining a direction, inclines our very inquiry in that direction" (128). Donna Haraway points out the concomitant militarism associated with informationalism: "like all neuroses, mine is rooted in the problem of metaphor, that is, the problem of the relation of bodies and language. . . . Technoscience and science fiction collapse into the sun of their radiant (ir)reality – war" (185). John Johnston explains how machines infuse post-Freudian psychoanalysis, arguing "the most salient fact about Lacan's *Seminar on the Ego* is that it is elaborated in relation to cybernetics and information theory," and that Lacan "simply implements in a functional model the most up-to-date media of information storage, transmission and computation" (23).

Turkle reminds us that Herbert Simon predicted a program model would be developed for psychological theory: "the new way of knowing asks that you think about everything, especially all aspects of the mind, in computational terms, in terms of programs and information processing" (*Second Self* 244). A computer presence as sustaining myth for the new cognitive science psychology of inner states is based on logic and rules. She goes as far as to suggest that the modernist computational aesthetic actually grounds postmodernism, which I touched on in the first chapter. "The image of the computer as calculator suggested that no matter how complicated a computer might seem, what happened inside it could be mechanically unpacked. . . . Computational ideas were presented as one of the great modern metanarratives, stories of how the world worked that provided unifying pictures and analyzed complicated things by breaking them down into simpler parts" (*Life on the Screen* 17).

Posthuman cybernetics feed on technological objects as the aims of research and development, the foundation of metaphors about the mind, and finally as objects to think with, our second self. Paul N. Edwards, in *The Closed World: Computers and the Politics of Discourse in Cold War America*, credits Turkle for showing “the analogy between computers and minds can simultaneously decenter, fragment, and reunify the self by reformulating self-understanding around concepts of information processing and modular mental programs, or by constituting an ideal form for thinking toward which people should strive. . . . With the emergence of global computer networks and virtual reality technologies for creating and inhabiting elaborated simulated spaces, by the 1990s cyberspace became a reality" (19-20). To him computer metaphors now pervade everyday self-understanding. Turkle claims computing's philosophical manifestation is artificial intelligence research; “in asserting the primacy of program, artificial intelligence is making a big claim, announcing itself, as psychoanalysis and Marxism had done, as a new way of understanding almost everything” (*Second Self* 246).

Turkle gives the example of Donald Norman's new take on Freudian slips as meaningless glitches in the mechanism, a view seconded by Marvin Minsky. “Norman believes that since the representations of knowledge developed for computers are efficient and rational, the evolution of human mental mechanisms is likely to have converged along similar lines. . . . A computational perspective suggests that when we understand the mechanisms of information storage and the programs that control our actions, we will come to see certain slips as likely, so likely that a Freudian search for meaning will be unnecessary and misguided” (*Second Self* 248). Without mentioning von Neumann's provocative statement that the best way to understand complex automata is to study machines rather than human beings, B. Jack Copeland proposes a new version of the Socratic 'know thyself' to the extent the brain can be assigned a computational interpretation. “It is always an empirical question whether or not there exists a labeling of some given naturally occurring system such that the system forms an honest model of some architecture-algorithm specification. And notwithstanding the truism that 'syntax is not intrinsic to physics' the discovery of this architecture-algorithm specification and labeling may be the key to understanding the system's organization and function” (357).[22]

Turkle contends that “under pressure from the computer, the question of mind in relation to machine is becoming a central cultural preoccupation. It is becoming for us what sex was to the Victorians threat and obsession, taboo and fascination” (*Second Self* 313). According to Wendy Chun, computers are now the metaphors for all effective procedures, the invisible generating visible effects; they are metaphors for stability, a strange stability arising from their ephemerality. “The linking of rationality with mysticism, knowability with what is unknown, makes it a powerful fetish that offers its programmers and users alike a sense of empowerment, of sovereign subjectivity, that covers over barely—a sense of profound ignorance" (*Programmed Visions* 17-18). Yet returning to Hayles' study of the Macy Conferences, it is important to remember that these metaphors are grounded in their particular social, cultural, ideological milieu.

For a time, rejecting cybernetic, embodied brain models enshrined an Enlightenment, closed mind, as Edwards relates. His closed world metaphor derives from the feedback control model as a dome of global oversight, epitomized by the US and Soviet programs to protect each other from intercontinental missile strikes. “As machine, computer controlled vast systems of

22 Turkle argues that interdisciplinary borrowing was necessary for AI to ground itself, but soon viewed it as colonization. Her position fits well with Hayles' studies, and it is also fitting that Floridi devotes such a large part of *Philosophy of Information* to AI.

military technology central to the globalist aims and apocalyptic terms of Cold War foreign policy. . . . As metaphors, such systems constituted a dome of global technological oversight, a *closed world*, within which every event was interpreted as part of a titanic struggle between the superpowers. . . . Computers made the closed world work simultaneously as technology, as political system, and as ideological mirage" (1).

Ironically, the closed world model inhibits artificial intelligence research by insisting that intelligent machines do *not* deploy artifice, revealing tensions between acknowledging embodiment, and rampant biochauvanism in early Lyotard and Baudrillard. "Artificial intelligence is devoid of intelligence because it is devoid of artifice. True artifice is the artifice of the body in the throes of passion, the artifice of the sign in seduction, the artifice of ambivalence in gesture, the artifice of the pithy remark that completely alters meaning. So-called intelligent machines deploy artifice only in the feeblest sense of the word, breaking linguistic, sexual, or cognitive acts down into their simplest elements and digitizing them so that they can be resynthesized according to models. They can generate all the possibilities of a program or of a potential object" (*Transparency of Evil* 52).[23] Edwards counters this one-sided view of cyborg identity by making a move similar to Hayles, teasing out the social and cultural contingencies of the closed world mentality in order to suggest alternate future trajectories. "In rejecting the cybernetic brain models and learning machines, AI also rejected a model of mind as inherently embodied. . . . In its Enlightenment-like proposals for an encyclopedic machine, AI sought to enclose and reproduce the world within the horizon and the language of systems and

23 We could compare different bases of reason and logic to contestations in Macy conferences highlighted by Edwards and Hayles. See Janz's "Reason and Rationality in Eze's *On Reason.*"

information. Disembodied AI, cyborg intelligence as formal model, thus constructed minds as miniature closed worlds, nested within and abstractly mirroring the larger world outside” (255-256). The model of mind deriving from closed world assumptions is out of sync with the as-is situation, in which we have been posthuman for a long time, but nonetheless embodied as well.

Cyborg Embodiment

My review of Hayles' effort to complicate and call into question our default assumptions about cybernetics and the ensuing model of the human nervous system that likens it to computer circuitry, while neuronal networks in computer circuity are likened to human brain functions, and human mental habits are described as programs, while psychotherapy is likened to a form of reprogramming, is intended to bring into question technological determinist metanarratives behind humanities criticism of the as-is condition, such as the ones by Weizenbaum, Bauerlein, and Winner with which this dissertation began. According to Hayles, Varela bridges the second and third waves of cybernetics as he split with Maturana over frustration with the inability of the latter's closed systems to deal with dynamic transformations between living systems and their environments. She points to Valera's work on enaction, which “sees the active engagement of an organism with the environment as the cornerstone of the organism's development. . . . In *The Embodied Mind: Cognitive Science and Human Experience*, Varela and his coauthors take the Buddhist-inspired point of view that the self is a story consciousness tells itself to block out the fear and panic that would ensue if human beings realized there is no essential self. . . . No longer Wiener's island of life in a sea of entropy or Maturana's autonomous circularity, awareness realizes itself as part of a larger whole unbounded, empty, and serene” (155).

Hayles points to Rodney Brooks' robot subsumption architecture as exemplary of a philosophy of embodiment that minimizes the criteria required for selfhood and subjectivity. “Whereas Moravec privileges consciousness as the essence of human being and wants to preserve it intact, Brooks speculates that the more essential property of the human being is the ability to move around and interact robustly with the environment. . . . There is no central representation, only a control system that kicks in to adjudicate when there is a conflict between the distributed modules. Brooks points out that the robot does not need to have a coherent concept of the world; instead it can learn what it needs directly through interaction with its environment. The philosophy is summed up in his aphorism: The world is its own best model” (*How We Became Posthuman* 235). In his later work Lyotard asserts that human thought is reflective and analogical, not determinate, logical and binary, despite imposed computational models, due to the debt owed to perceptual experience and embodiment in general (*The Inhuman* 15-16). Bringing the body and environment into prominence should also alter the balance between purely logical, symbolic characterizations of cognition, and those that take into account emotional and haptic phenomena. With third wave cybernetics and contemporary human brain science aligning on massively parallel, fuzzy logic networks as the best model for the mind, the physical characteristics of the interface between organism and highly engineered environment become relevant characteristics of the formerly monolithic, Cartesian ego that grounded the humanities self-conception during the epoch characterized by stored images of written text and mechanically-effected media.

Next I will explain how Lucy Suchman's development of situated actions articulates her critique of Cartesian reasoning from the standpoint of computer science, followed by how Andy

Clark's extended mind builds upon McLuhan's declaration that the human nervous system extends into electric media not merely as perceptual stimulus and response but also its computational, even intellectual partner. Together they provide an adjustment to the depiction of the as-is situation that landed on the overly discursive subject of posthuman cybernetics.

Suchman begins her critique of traditional, 'good old fashioned', cognitive science based AI research by comparing the accounts of European and Trukese navigation. The former use charts, take readings from the stars and other instruments, plot their course, and then sail in the computed direction. The latter point their vessel towards their destination, begin sailing, and continuously make small adjustments as they proceed. Her point is that we act like Trukese while talking like Europeans. The view of action exemplified by the European navigator has been reified in the design of intelligent machines, in which purposeful action is determined by plans as the correct model of the rational actor. She cautions that “as students of human action we ignore the Trukese navigator at our peril. . . . My central concern in the investigation is a new manifestation of an old problem in the study of mutual intelligibility: namely, the relation between observable behavior and the processes, not available to direct observation, that make behavior meaningful” (*Plans and Situated Actions* x; 2).

The error in AI work is creating systems based on planning models, for they confuse plans, the processes largely hidden and retrospective, with situated actions, the observable, immediate behavior. Suchman invokes Turkle's notion of the computer as an evocative object, “one that raises new questions regarding our common sense of the distinction between artifacts and intelligent others” (5). In particular, they provoke scientists and humanists alike to criticize the conflation of shared understanding and interaction, as interaction now includes programmed machines whose intentions are concealed in microscopic electronic circuits. Indeed, the interactivity of computational artifacts is actually supported by their reactive, linguistic, internally opaque properties. Echoing Turkle's pronouncement about the quintessential postmodern object, user interfaces are designed to work on the surface level. Suchman finds that this functionalism at the heart of cognitive science is based on de la Mettrie's view of mind as an abstractable structure implementable in many substrates, and it entails that computer modeling, as information processing psychology, ought to provide accountability to pursue any science of inaccessible mental phenomena.

The first, and most questionable, premise that it makes is that all cognizers act on the basis of symbolic representations, and the second, that this cognitive code is instantiated physically in the brain (9). Hence many believe artificial intelligence research should be done like European navigation, pointing to the success of commercial expert systems and industrial robots. Suchman illuminates the outcome of this approach. “In both cases, the systems can handle large amounts of encoded information, and syntactic relationships of great sophistication and complexity, in highly circumscribed domains. But when it comes either to direct interaction with the environment, of to the exercise of practical, everyday reasoning about the significance of events in the world, there is general agreement that the start-of-the-art in intelligent machines have yet to attain the basic cognitive abilities of the normal five-year-old child” (10).[24]

24 This dated view of AI research is meant to reflect popular opinion during the lead up to the Internet epoch, not report on the state of the art, which indeed has taken many of the recommendations made by Suchman to heart.

Despite their poor general cognitive abilities, computer systems have given many people the impression that there is some intelligence behind their behavior. Reactive, interactive computers can be purposeful, social objects. Because machine control has become more linguistic than mechanical, Suchman reasons that “with or without machine intelligence, this fact has contributed to the tendency of designers, in describing what goes on between people and machines, to employ terms borrowed from the description of human interaction dialogue, conversation, and so forth: terms that carry a largely unarticulated collection of intuitions about properties common to human communication and the use of computer-based machines” (11). Furthermore, just as humans lean on intentional explanations for interpreting each others' actions due to the opacity of individual perspective, “it is in part the internal complexity and opacity of the computer that invites an intentional stance. . . . To refer to the behavior of the machine, then, one must speak of its functionality” (15-16). Suchman adds two ingredients to the mix to explain the psychological phenomenon that so troubled Weizenbaum about people's acceptance of his DOCTOR program and other conversational computer systems. The Watt habitability problem leads to assuming sophisticated linguistic abilities after witnessing elementary ones, and the Mannheim documentary method of interpretation explains why people believe there is artificial intelligence behind it. “Very simply, the documentary method refers to the observation that people take appearances as evidence for, or the document of, an ascribed underlying reality, while taking the reality so ascribed as a resource for the interpretation of the appearance" (23).

Contrary to the false conclusions reached by these two tendencies about conversational computer systems, Suchman aligns with Terry Winograd, who declares the computers of their time are language machines, not thinking machines. “The computer is a physical embodiment of the symbolic calculations envisaged by Hobbes and Liebniz. As such, it is really not a thinking machine, but a language machine. The very notion of symbol system is inherently linguistic, and what we duplicate in our programs with their rules and propositions is really a form of verbal argument, not the workings of mind. . . . artificial intelligence has operated with the faith that mind is linguistic down to the microscopic level" ("Thinking Machines" 216-217).[25] An early stage of cyborg embodiment found humans equipped with writing skills, but it in turn led to conceptions of the mind and cognition that favored discursivity.

Neel reflects that "Plato operates in a world of writing in which what one knows emerges through the recursive, nonsequential externalizing of discourse, discourse that then remains available for limitless modification. At the same time, he valorizes a world in which knowledge is an entirely internal phenomenon that, rather than being created, is gradually recognized through oral discourse with a lover" (*Plato Derrida Writing* 38). The turn in decent decades has been to "advance the idea of technical media as an autonomous force determining subjectivity," Hayles contends, singling out Kittler as the leading proponent; "influenced by Foucault rhetorically as well as methodologically, Kittler departs from him in focusing not on discourse networks understood as written documents, but rather on the modes of technology essential to their production, storage, and transmission" (*Electronic Literature* 88-89). Kittler extends the logocentrism of Platonic writing into the human organism by featuring the 'invention' of internal

25 This assumption that the mind is linguistic all the way down is precisely the point so many find defective in the OHCO thesis about textuality and the one Hayles exclaims overdetermines early posthuman conceptions as inherently discursive and symbolic.

speech as a prime instance of technical media determining subjectivity by actualizing literary language in the body.

Introduced by Garret Stewart, subvocalization occurring during silent reading is essential to the production of literary language. As Hayles explains, “language becomes literary for Stewart when it cannot be adequately replaced by other words, when that particular language is essential to achieving its effects. Literary language works by surrounding its utterances with a shimmer of virtual sounds, homophonic variants that suggest alternative readings to the words actually printed on the page. Subvocalization actualizes these possibilities in the body and makes them available for interpretation" (*How We Became Posthuman* 207). Subvocalized voice replaces material grapheme with auditory hallucination. Kittler contends that subvocalization emerged in the discourse networks of the 1800s through the Mother's voice, as women taught their children to read by reading to them. As noted at the beginning of the chapter, phonetic reading is literally text-to-speech synthesis, like playing from a musical score. Human subjectivity transformed along with this technology to the point that it, too, seems natural.

It is beyond the scope of this work to delve into the mysteries of neurophysiology and the mechanics of cognition.[26] However, an excursion into Andy Clark's theory of extended cognition serves to present a plausible middle ground between accounts of the mind that reduce it to a homuncular singularity (the Cartesian soul) or a symbol processing, natural automaton (the cybernetic mind). For the purpose of describing the as-is situation of the post-postmodern human subject, this approach is well equipped to acknowledge the increasing role of the device-laden

26 Hayles uses Mark B. N. Hansen as foil to Kittler's technological determinism by foregrounding the role of embodiment in perceptual experience.

built environment, which will be discussed as code/space in chapter three, as an integral part of the perceptual and cognitive apparatus. Extended cognition does not require a technologized environment to manifest itself, however. Clark gives the example of the Bluefin tuna: “the physical system whose functioning explains the prodigious swimming capacities of the Bluefin tuna is thus the fish-as-embedded-in, and as actively exploiting its local environment” (“Embodied Cognitive Science” 345).

Like Brooks' subsumption architecture Hayles discussed in the context of reduced requirements of selfhood deprivileging consciousness toward a more holistic, systems perspective, Clark contends that an ideal robotic controller ought to exploit intrinsic dynamics of the environment like the Bluefin tuna. Thus, the task of robotic vision must be toward efficiency in service of real-world, real-time action rather than building rich inner models. Likewise, traditional models of cognition imply that the brain produces a richly detailed model of perceptual phenomena, and then reasons based on that virtual representation. He invokes J. J. Gibson's concept of affordances from ecological psychology, which challenges traditional ideas about perception and action assuming rich internal representations with an adaptively potent equilibrium coupling agent and world. “This approach stresses bodily movement, ecological context and the action-relevant information available in the perceptual array. . . . They replace the notion of rich internal representations and computations, with the notion of less expensive strategies whose task is not first to represent the world and then reason on the basis of representation, but instead to maintain a kind of adaptively potent equilibrium that couples the agent and the world together” (“Embodied Cognitive Science” 346). Cognitive capabilities expand as organisms incorporate new, fine motor control behaviors, such as the use of hands to build and use tools.[27]

The tendency to slip back into a dichotomy of user and tools is easy. As David Chalmers, a long time collaborator of Clark, explains in the Foreword to *Supersizing the Mind*, the limited bandwidth between elements of environment allows maintaining an internal conscious core for humans; at the same time, accepting the weak functionalism of the Parity Principle supports mentality existing in the cyborg network (xv). Clark presses for acknowledgment that “one way in which evolved agents truly inhabit rather than simply control, their bodies may be usefully understood in terms of a profound fit between morphology and control” (10-11). This fit between morphology and control is at the core of multipurposive synergetic systems. A Cartesian, modernist subjectivity separating perception and cognition as manipulating internal models is being supplanted by an open conduit; “*the embodied agent is empowered to use active sensing and perceptual coupling in ways that simplify neural problem solving by making the most of environmental opportunities and information freely available in the optic array*” (17).[28]

The hybrid stance Clark lands on is informed by more recent models of computing than earlier cybernetics, tying in Hayles' third wave and her later focus on intermediation in *Electronic Literature*. “Instead, what we seem to end up with is a very powerful and interesting

27 Hayles makes a similar point concerning use of electronic literature in the context of synaptogenesis.

28 Clark suggests a polarization of colliding philosophies at the heart of the study of subjectivity and artificial intelligence: criticism of Dynamical Systems Theory from computer science and process control engineering, traffic and routing problems of multiprocessing and distributed systems that were not imagined by early computing theorists grounded in Turing machines and single processing von Neumann architectures.

hybrid: a kind of dynamical computationalism in which the details of the flow of information are every bit as important as the larger scale dynamics and in which some dynamical features lead a double life as elements in an information-processing economy. . . . Such work aims to display the specific contributions that embodiment and environmental embedding make by identifying what might be termed the dynamic functional role of specific bodily and worldly operations in the real-time performance of some task" (27).

A consequence of the extended position is that it makes it easier to conceive of the mind as a collection of heterogeneous, disunified processes. Hayles refers to theories by Jackendoff and Minsky for showing that "contemporary models of cognition can be modeled through discrete and semiautonomous agents. Each agent runs a modular program designed to accomplish a specific activity, operating relatively independent of the others. Only when conflicts occur between agents does an adjudicating program kick in to resolve the problem. In this model, consciousness emerges as an epiphenomenon whose role it is to tell a coherent story about what is happening, even though this story may have little to do with what is happening processurally" (*How We Became Posthuman* 156-157). The processural model of disunified processes readily accepts not just inputs and outputs coupled beyond the boundaries of the biological entity, but continuations of these subsystems.

A key transition for articulating cyborg embodiment is whether cognitive extension is a meaningful concept like empirically documented sensory extension. Clark argues "to insist that such change requires the literal intelligibility of the operations of the new by the old, rather than simply the emergence for new wholes that are then *themselves* the determiners of what is and is not intelligible to the agent. It must thus be possible, at least in principle, for new *nonbiological* tools and structures to likewise become sufficiently well integrated into our problem-solving activity as to yield new agent-constituting wholes" (40). He refers to change blindness experiments strengthening the idea that embodiment extended into the environment sustains reliance on the visual field in place of constant inspection. That is, we rely on the ready-at-handedness of visual phenomena to allow ourselves to sample rather than maintain constant attention.

Our reliance on ready-at-handedness extended to wearable computing and ubiquitous information access likewise strengthens the notion of cognition extended into the environment, including non-biological components. “Whereas BRAINBOUND locates all our mental machinery firmly in the head and central nervous system, EXTENDED allows at least some aspects of human cognition to be realized by the ongoing work of the body and/or the extraorganismic environment” (82). Clark presents the clearest, most sensible, up to date basis for the philosophy of computing as it intersects mind, cognition, consciousness, subjectivity and ultimately the human. His themes of spreading the load, self-structuring of information, supporting extended cognition joined by his hypotheses of cognitive impartiality, motor deference, multiple functionality are all highly relevant to articulating cyborg embodiment.[29]

29 Physical underpinnings are well explained by this model of subjectivity, raising questions to be entertained as we consider machine embodiment: given its proximity to machine systems, does it follow that machine cognition independent of human may exist; need it be disconnected to be genuine; can the humans be parts of incomprehensible, alien swirls for it?

Clark and Chalmers argue further that “*beliefs* can be constituted partly by features of the environment, when those features play the right sort of role in driving cognitive processes. If so, the mind extends into the world” (“Extended Mind” 12). They also suggest the notion of socially extended cognition: “Could my mental states be partly constituted by the states of other thinkers? We see no reason why not, in principle” (17). They often use a device metaphor for describing unique abilities of language, here as ubiquitous literacy. "Without language, we might be much more akin to discrete Cartesian 'inner' minds, in which high-level cognition relies largely on internal resources. But the advent of language has allowed us to spread this burden into the world. Language, thus constructed, is not a mirror of our inner states but a complement to them. It serves as a tool whose role is to extend cognition in ways that on-board devices cannot" (18). On the one hand, it seems improper to describe these transformations as cybernetic because they emerged accidentally, not as part of a deliberately engineered solution. On the other hand, Clark contends that “our mature mental routines are not merely self-engineered: They are massively, overwhelmingly, almost *unimaginably* self-engineered. The linguistic scaffoldings that surround us, and that we ourselves create, are both cognition enhancing in their own right and help provide the tools we use to discover and build the myriad *other* props and scaffoldings whose cumulative effect is to press minds like ours from the biological flux” (*Supersizing the Mind* 60). They are the quintessence of what Engelbart calls Type C activity, improving improvement.[30]

30 There is regular activity, for example office work (Type A), activity to improve activity, for example office automation (Type B), and activity to improve improvement (Type C). The goal of his augmentation research lab was to develop tools and practices that bootstrapped their own improvement, such as distributed graphical desktop computing applications that were used to further develop those very tools.

A final step in this development, before flipping over to look at the networks enveloping human beings and the planet itself, is to further differentiate cyborg embodiment from the largely discursive subjectivity characteristic of postmodernism that was presented in the previous section. Hayles refers to Lakoff and Johnson's study of embodied metaphors, which demonstrate how “our images of our bodies, their limitations and possibilities, openings and self-containments, inform how we envision the intellectual territories we stake out and occupy” (*How We Became Posthuman* 85). It is not merely that abstract concepts in languages are based on embodied metaphors; rather, a significant portion of cognitive activity operates outside the discursive, symbolic realm. Her aim is to complicate the equivocation between human cognition and computer programs. Valera and Dreyfus emphasize emotion and other embodiment-dependent aspects of learning and intellection in addition to rational cognition (201).

Hayles develops their argument by foregrounding distinctions between inscription and incorporation. “Like the body, inscription is normalized and abstract, in the sense that it is usually considered as a system of signs operating independently of any particular manifestations. . . . I foreground them now to point out that they constitute inscription as a conceptual abstraction rather than as an instantiated materiality. . . . In contrast to inscription is incorporation. An incorporating practice such as a good-bye wave cannot be separated from its embodied medium, for it exists as such only when it is instantiated in a particular hand making a particular kind of gesture” (198). The good-bye wave is a habit that is not symbolizable; it cannot be readily stored in symbols. Dreyfus, who was mentioned earlier, differentiates embodied learning from computer programs because activities like the hand waving include an inner horizon that is only partly determined, whose meaning is deferred in the partly open context of anticipation. Moreover, the anticipation takes on a global character that crosses other sense modalities (201).

Hayles also invokes Pierre Bourdieu, the French sociologist, who “argues that even if one is successful in reducing some area of embodied knowledge to analytical categories and explicit procedures, one has in the process changed the kind of knowledge it is, for the fluid, contextual interconnections that define the open horizon of embodied interactions would have solidified into discrete entities and sequential instructions” (202). Bourdieu's concept of *habitus* exemplifies embodied knowledge independent of discursive articulation. "Bourdieu's work illustrates how embodied knowledge can be structurally elaborate, conceptually coherent, and durably installed without ever having to be cognitively recognized as such. . . . The habitus is conveyed through the orientation and movement of the body as it traverses cultural spaces and experiences temporal rhythms" (202-203). Likewise, post-postmodern cyborg embodiment challenges both premises, that cognition is localized in the human brain, and that it is linguistic all the way down, by extending the mind beyond the human body into the environment, and by acknowledging multiple modes of cognition, adding Bruner's enactive – like the Bluefin tuna – and iconic – like the good-bye wave – mentalities to the strictly symbolic.

Techno-capitalist Networks

Flipping the focus from the extended mind to the machine side I seek to sketch out their milieu, which I call techno-capitalist networks. This gives form to that which other theorists have called biopower, bureaucratic machine, built environment, global intelligence, Leviathan, and so on. Winner notes that “a consistently ahistorical viewpoint prevails. . . . Politics, in other words, is not a secondary concern for many computer enthusiasts; it is a crucial albeit thoughtless, part of their message" ("Mythinformation" 590). Yet the reality, he argues, is that the benefits accruing to individuals are minuscule in comparison to those large, transnational businesses and governmental institutions reap from planetary technologicalization. "While their global reach does not arise solely from the application of information technologies, such organizations are uniquely situated to exploit the efficiency, productivity, command, and control the new electronics make available. Other notable beneficiaries of the systematic use of vast amounts of digitized information are public bureaucracies, intelligence agencies, and an ever-expanding military, organizations that would operate less effectively at their present scale were it not for the use of computer power" (592).

As we consider the as-is situation of ordinary individuals, whose posthuman minds and bodies extend beyond the boundaries of their flesh and bones, the omnipresence of these collective entities—despite their putative immateriality—cannot be ignored. Deleuze's famous "Postscript on the Societies of Control" posits a universal system of deformation to define the milieu from which techno-capitalist networks emerge. "The disciplinary man was a discontinuous producer of energy, but the man of control is undulatory, in orbit, in a continuous network" (6). His grand statement about control versus confinement reflects continuous industrial cybernetic feedback process control enveloping humanity. "It's a capitalism of higher-order production. It no longer buys raw materials and it no longer sells the finished products. What it wants to sell is services and what it wants to buy is stocks. . . . The operation of markets is now the instrument of social control and forms the impudent breed of our masters. Control is short-term and of rapid rates of turnover, but also continuous and without limit, while discipline was of long duration, infinite and discontinuous" (6).

Prior to Deleuze, Foucault observed that indefinite discipline under endless interrogation is the model of the control society, asking the rhetorical question "Is it surprising that prisons resemble factories, schools, barracks, hospitals, which all resemble prisons?" (*Discipline and Punish* 227-228). As he picked apart the mechanics of the internal regulation of governmental rationality, he foreshadowed the contemporary, technical concept of protocol. "To say that there is a de facto limitation of governmental practice means that a government that ignores this limitation will not be an illegitimate, usurping government, but simply a clumsy, inadequate government that does not do the proper thing. . . . Internal regulation means that there really is a limitation that is general while being de facto, that is to say, that, whatever happens, follows a relatively uniform line in terms of principles valid at all times and in all circumstances" (*Birth of Biopolitics* 10-11). Foucault claims that his studies of madness, disease, delinquency, and sexuality all illustrate "how the coupling of a set of practices and a regime of truth form an apparatus (*dispositif*) of knowledge-power that effectively marks out in reality that which does not exist and legitimately submits it to the division between true and false" (19). Yet he contends that the core aim of these apparatus are to create "this new type of rationality in the art of government, this new type of calculation that consists in saying and telling government: I accept, wish, plan, and calculate that all this should be left alone," and its name is liberalism (20).

Foucault discerns in the development of the capitalist market as the site of veridication-falsification for governmental practice the mechanism underlying all sorts of modern power networks. This is a reversal of the mythical, heavy-handed rule of monarchies who consolidated governmental power in order to further their own ends. Liberalism seeks instead to govern in order to allow interests to seek out their aims. “Government is only interested in interests. . . . [I]t deals with interests, or that respect in which a given individual, thing, wealth, and so on interests other individuals or the collective body of individuals. . . . The insertion of this thin phenomenal film of interests as the only sphere, or rather, as the only possible surface of government intervention, is what explains these changes, all of which must be referred back to this reorganization of governmental reason” (45). The focus of governmental action shifts from individuals to networks of relations, ultimately appealing to exchange. “In its new regime, government is basically no longer to be exercised over subjects and other things subjected through these subjects. Government is now to be exercised over what we could call the phenomenal republic of interests. The fundamental question of liberalism is: What is the utility value of government and all actions of government in a society where exchange determines the true value of things?” (46). Biopower and capitalism are fundamentally connected, even if the former seems to point to socialism instead, due to their underlying principles of operation that Foucault's research helped reveal.[31]

31 Foucault's mention of the obsession Frederick II had with automata foreshadows their concretization into the modern bureaucratic state apparatus. Indeed, an amusing story may be told of various famous philosopher's obsessions with humanoid automata, notably Descartes' mysterious companion Francine noted by Sterne: “Cartesianism introduces mechanism into modern philosophy. But Descartes's interest in mechanism went further—he was, in fact, fascinated with automata. Price reports rumors that Descartes planned to build a dancing man, a flying pigeon, and a spaniel that chased a peasant and that he did build a blonde automaton named Francine that was discovered in her packing case aboard a ship and summarily thrown overboard by a

Capitalism has been invoked many times as the fundamental metaphysic behind our modern era. Luc Boltanski and Eve Chiapello, in *The New Spirit of Capitalism*, describe it as unlimited, interminable accumulation of capital by formally peaceful means, distinguished from the market economy that attempts to regulate it (4). Capitalists formally include those possessing property income, and capitalism is also characterized by voluntary subjection of a wage earning class, but their study focuses on active profit maximizers of commercial firms. *Cadres*, meaning 'career plans', embody a key concept for the limit of the multitude of little people, middle management, bordering the great men, top executives, epitomizing the new spirit of capitalism in humans. What they call the 'spirit of capitalism' is the ideology justifying engagement in it (8). Following Weber, vocation has been treated as a moral relationship to work, the famous Protestant ethic. To Hirschman, Enlightenment secular thinking justified profit making in terms of common social good; lucre became innocuous passion for subjugating more aggressive passions (9-10). Utilitarianism incorporated into economics connected profit creation with a common good serving society. Commoditization of services by competitive private enterprise is deemed a socially optimal solution because of its dual drive to maximize profit through reducing waste and to satisfy customers, including anticipating their expectations (13).

Importantly, the spirit of capitalism peculiar to each age must assuage anxiety provoked by questions of autonomy, security, and the common good (16). The first characterization of capitalism focused on the bourgeois entrepreneur at end of nineteenth century; its amalgam of incompatible propensities led to charges of hypocrisy (17). The second characterization developing between 1930s and 1960s emphasized the organization and the heroic manager.

captain frightened of witchcraft" (72-73).

Moreover, control transferred to the technostructure (17-18). At this point their phased account of capitalist development, accompanied by different spirits of engagement, begins to dovetail Hayles' history of cybernetics. This stage parallels the Cold War, closed world paradigm. The third characterization of globalized capitalism employing new technologies reflects the ideological world picture of free Western European and United States versus planned economies of 1960s, only to be replaced by the emergence of a third capitalist pole in Asia challenging old capitalist countries in the 1990s (72).

In the new spirit of capitalism lean, network, projects, vision, alliance, team are keywords (73). Increase in productivity of investments drives replacement for the Fordist mode of regulation, accompanied by the economic importance of worker awareness of good health of the machines (81-82). Replacing hierarchical control with market control by outsourcing and autonomization of the value stream focused on the customer's order is hard to plan, and depends on the ability to react, organizational flexibility, and trust. Thus business theorists conclude that bureaucracies are not only inhuman but unviable, and contemporary management rhetorics reintroduce personal relations to the inhuman machines created by endeavors to relentlessly rationalize firms (85; 87).

Boltanski and Chiapello identify a weak point in this new spirit of capitalism when proposing forms of security compatible with the dominant requirement of flexibility (89). Indeed, security is offered for cadres in the form of career guarantees, and the guarantee of a welfare state for everyone else (87). Those who pursue lifelong learning and are willing to cut their social and regional ties to remain in the job market, moving from project to project, floating on rafts of opportunity in the welfare state. The third, new spirit of capitalism reacts to the closed world logics of business necessity that created the informational economy. “The 1990s hark back to this idea in order to counter it: the endeavor relentlessly to rationalize the way firms are run has created inhuman machines” (87). Critical sociology of the 1990s and onwards has turned its attention to inhuman affairs and their relationship to global capitalism. Nicholas Gane argues that “unlike McLuhan, the underlying message here is quite different: these technologies are not merely 'extensions of man' but extensions of the capitalist market insofar as they speed up and thus promote the production, exchange and consumption of information” (“Computerized Capitalism” 431-432).

Boltanski and Chiapello derived their new spirit of capitalism from analyses of keywords and themes of popular management texts from the 1940s to the present. While the scope of their study was limited to France, most of the texts originated in United States schools of business administration, sociology, and psychology departments. They tell the story of the transformation of capitalist, transnational business tracking the development of cybernetic information technologies. Manuel Castells dubbed it the rise of the network society, producing a multi-volume study based on a worldwide research trip he conducted with his colleague Peter Hall beginning in 1988. The network enterprise is constituted by business networks, technological tools, global competition, and the state. What happens to human beings inhabiting it is that the network replaces subjectivity as basic unit of economic organization, actor, or agent (*Rise of the Network Society* 214). The network enterprise is the material, productive instantiation of the informational culture, “t*hat specific form of enterprise whose system of means is constituted by the intersection of segments of autonomous systems of goals*. . . . The performance of a given network will then depend on two fundamental attributes of the network: its *connectedness*, that is, its structural ability to facilitate noise-free communication between its components; and its *consistency*, that is, the extent to which there is a sharing of interests between the network's goals and the goals of its components. . . . In this sense, *the network enterprise makes material the culture of the informational, global economy: it transforms signals into commodities by processing technology*" (187-188).

Paralleling Hayles' narrative of how information lost its body and we became posthuman, the faceless collective capitalist of networked financial flows replaces the iconic, sovereign leader. "Thus, above a diversity of human-flesh capitalists and capitalist groups there is a faceless collective capitalist, made up of financial flows operated by electronic networks." Castells continues that "the new social order, the network society, increasingly appears to most people as a meta-social disorder. Namely, as an automated, random sequence of events, derived from the uncontrollable logic of markets, technology, geopolitical order, or biological determination" (505). The as-is situation includes this sense of a built environment transformed from physical to informational metaphysics, mirroring the transformation of subjectivity presented in the first section of this chapter. "We are just entering a new stage in which culture refers to culture, having superseded nature to the point that nature is artificially revived (preserved) as a cultural form: this is in fact the meaning of the environmental movement, to reconstruct nature as an ideal cultural form" (508). Seen this way, the machine environment reflects human art, artifacts, designs, devices, materials, handicrafts and so on because it has spent the orality and literacy eras servicing us. That somehow the semblance of an individual, conscious, intentional intelligence exists in that mass of independent, distributed computing systems in a like manner must be explicated to articulate the as-is situation on the *inhuman* aspect of post-postmodern network dividual cyborg existence.

Boltanski and Chiapello ask us to consider the influencing power of concentrated assets on markets as an example of global capitalism expanding “without taking the form of investment in productive activity. . . . The liquid assets concentrated in the hands of mutual investment funds, insurance companies and pension plans are such that their capacity to influence the markets in accordance with their interests is a recognized fact” (xxxvi). I suggest that we consider this phenomenon as expressive of a type of collective, machinic cognition in which the cumulative reckoning of encoded financial information triggers semi-autonomous and autonomous feedback responses that literally move markets. The neural network model from cybernetics contributes to explain how countless, interconnected, unintelligent nodes operate together to produce the semblance of plans from the conjunction of billions of situated actions.

As Boltanski and Chiapello relate, the network model has also taken over the ontology of the business world. “Large firms have not been dissolved into a set of commercial contracts between small units competing in a pure, perfect atomized market . . . networks constitute a specific form between hierarchies and markets,” and moreover, “the answers proposed by 1990s management literature to the two questions of most concern to it—anti-authoritarianism, and an obsession with flexibility and the ability to react—are conveniently assembled by the authors under the *metaphor of the network*, which is deployed in all sorts of contexts” (84). It is important to note that this term has been appropriated from other contexts and mobilized by the rhetoric of capitalism. “As an existing concept, constructed around contemporary ideas, technologies and research, associated with a specific vocabulary, models of causality and mathematical models, and formed to offer an alternative to hierarchical algorithms, 'network' naturally enough finds itself mobilized by capitalism. Employed in academic works in economics and the sociology of work disciplines that helped to provide management with its theoretical foundations it was almost bound to invade the literature to *cadres* that we have studied” (104). Thus the network form in firms, hitherto associated with organized crime, has been rehabilitated and elevated to fundamental principle of organizational management (128). Networks appeal to desires to connect as basic property of human nature, as well as wanting to be simultaneously free and engaged.

According to Castells, two key features of the technological paradigm include attention to complexity theory's non-linear dynamics for understanding the behavior of living systems, and integrating the logic of biology into electronic machines, both features of the second and third waves of cybernetics that Hayles elaborates (72). “In sum, the information technology paradigm does not evolve toward its closure as a system, but toward its openness as a multi-edged network. It is powerful and imposing in its materiality, but adaptive and open-ended in its historical development. Comprehensiveness, complexity, and networking are its decisive qualities” (75-76). Directed to symbol processing, the information technology (IT) paradigm has brought about a fundamental change from a world economy to a global economy: “a global economy is something different: it is an economy with the capacity to work as a unit in real time, or chose time, on a planetary scale. . . . This globalized core includes financial markets, international trade, transnational production, and, to some extent, science and technology, and specialty labor” (101).

Moreover, the trading units of globalized markets are networks of firms rather than countries, “organized worldwide in their actual operating procedures, forming what Robert Reich labeled the global web. The production process incorporates components produced in many different locations by different firms, and assembled for specific purposes and specific markets in a new form of production and commercialization: high-volume, flexible, customized production. . . . What is fundamental in this web-like industrial structure is that it is territorially spread throughout the world, and its geometry keeps changing, as a whole and for each individual unit" (122-123). As Boltanski and Chiapello noted, economic theory has appropriated terms from other disciplines, so that 'web-like structure' and 'network' descriptively refer to this new global configuration, with self-reinforcing normative authority so that the network model is also heralded as the ideal. Castells notes that in network societies knowledge is primarily open, but research agendas are driven by the concerns of advanced countries. He also provides an interesting account of how governments, monetary funds, and trade organizations imposed pressure to adopt homogeneous rules of the game resulting in the open, global economy (140). Once accomplished, securitization liquidates all sources of value into common, tradable form, further loosening the flows of capital, so that “financial markets constitute the strategic, dominant network of the new economy” (156).

Castells offers an analysis yielding three explanatory layers for his network model in which familiar, physical space is reterritorialized by the field effect phenomena of networks, creating baseless structures that he calls the *space of flows*. “There follows the separation between symbolic meaning, location of functions, and the social appropriation of space in the metropolitan area. This is the trend underlying the most important transformation of urban forms worldwide, with particular force in the newly industrializing areas: the rise of mega-cities" (433-434). The first layer material support of the space of flows is constituted by circuits of electronic exchanges, the physical infrastructure of cyberspace. "Thus, the network of communication is the fundamental spatial configuration: places do not disappear, but their logic and their meaning become absorbed in the network" (443). He notes the virtuous circle of using IT to enhance IT, especially relevant in the example of Cisco Systems' meteoric corporate growth while it developed and supplied the basic infrastructure of new local and wide area computer communications networks. "The core of Cisco Systems operation is its web site. . . . Only major contracts are dealt with in person. . . . By networking its operation internally and externally, using the equipment it designs and sells, Cisco Systems epitomizes the virtuous circle of the information technology revolution: the use of information technologies to enhance the technology of information, on the basis of organizational networking powered by information networks" (182). In this example Cisco Corporation as a collective intelligence practices Engelbart's Type C activity of improving improvement.

The second layer of the space of flows "*is constituted by its nodes and hubs*. . . . Location in the node links up the locality with the whole network" (443). Humanities theorists have employed the term rhizome to describe them before, but they are materially enacted in computer networks, particularly in the predominant Transmission Control Protocol/Internet Protocol (TCP/IP) model. Understanding this specific technology that has become basis of metaphor will be a key ingredient to the procedural rhetoric of diachrony in synchrony I develop in the next chapter. Interestingly, the third layer of the space of flows shifts from the configuration of the technological infrastructure to "*the spatial organization of the dominant, managerial elites* (rather than classes) that exercise the directional functions around which such space is articulated. . . . Articulation of the elites, segmentation and disorganization of the masses seem to be the twin mechanisms of social domination in our society" (445). Organizational change has been viewed in America as saving labor and concentrating managerial control, independently of technological change, "as a response to the need to cope with a constantly changing operational environment. Yet, once it started to take place, the feasibility of organizational change was extraordinarily enhanced by new information technologies. . . . It was because of the networking needs of new organizations, large and small, that personal computers and computer networking underwent an explosive diffusion" (184). In a sense, global, collective intelligence absorbed the structures and efficiencies afforded by the space of flows, ushering in more fully the societies of control that Deleuze and others realized were replacing the disciplinary societies of the prior era.

This shift in the assumed locus of control is the foundation of Boltanski and Chiapello's new spirit of capitalism, in which the prior justifications of capitalist activity – and social action in general – grounded in avarice, industriousness, family, prestige, and so on, are replaced by lifelong participation in an endless progression of projects. Castells sees the transition to project-oriented business models an inevitable corollary to the rise of the network model: "*the actual operating unit becomes the business project, enacted by a network*, rather than individual companies or formal groupings of companies. . . . New information technologies are decisive in allowing such a flexible, adaptive model to actually work" (177). Consequently, the second finesse adopted by the rhetoric of global capitalism after redeeming networks from organized crime appropriates the term *project* as one of its key watchwords. "Projects make production and accumulation possible in a world which, were it to be purely connexionist, would simply contain flows, where nothing could be stabilized, accumulated or crystallized. . . . It is thus a temporary *pocket of accumulation* which, creating value, provides a base for the requirement of extending the network by furthering connections" (*New Spirit of Capitalism* 104-105). What sort of beings traverse networks participating in projects?

Dividual Cyborg

Consider the *post-postmodern network dividual cyborg* as a transitionary stage of human evolution on the way to the yet-to-be-named characters of the Axiom's passengers from *WALL-E*, if the trajectory traced in the previous sections concretizes as the new spirit of capitalism further extends into all aspects of humanity. This hypothetical characterization of subjectivity should be understood as a diagram of current, mainstream America, the little people of the projective city articulated by Boltanski and Chiapello. It connects the techno-evangelistic future predictions of Kurzweil's *Age of Spiritual Machines* with the more modest moralism of Rushkoff's *Program or Be Programmed*, by recognizing with Hayles, Edwards, and Golumbia among others the situated, contested histories of the dominant technologies whose current, default configurations are taken as inevitable outcomes of the progress of civilization. The human side of techno-capitalist networks being described now forces reflection upon the dividual described by Deleuze, Guattari, and others.

The subject becomes the *dividual*, the great man the nomad, and later, the cyborg, at its peaks, and little people everyone else: Boltanksi and Chiapello specify this interpretation of the situation. In Deleuze and Guattari's *A Thousand Plateaus*, the concept of the dividual is presented in the context of music appreciation, a one-crowd of orchestration-instrumentation, but in Deleuze's famous Postscript is spelled out in the way commonly associated with the human situation in the age of techno-capitalist networks. In the former, the authors are considering the oscillation between the one-as-all and the one-as-crowd. “In the first case, it is a question of effecting groupings of powers, and these are what constitute affects; in the second case, it is group individuations that constitute affect and are the object of orchestration. Groupings of power are fully diversified, but they are like the relations proper to the Universal; we must use another word, the Dividual, to designate the type of musical relations and the intra- or intergroup passages occurring in group individuation” (341). They add the disclaimer that “the concepts of the One-Crowd and the Dividual are botched if the people is reduced to a juxtaposition, or if it is reduced to a power of the universal,” and such reductions seem to be the case for the later characterization by Deleuze, where control has infiltrated everything, including putatively competing social alternatives, and “the corporation, the educational system, the armed services being metastable states coexisting in one and the same modulation, like a universal system of deformation” (5).

This notion of a universal system of deformation represents the techno-capitalist networks in which global humanity operates by the logic of codes and protocols, rather than individuals within a mass. “The disciplinary societies have two poles: the signature that designates the *individual*, and the number or administrative numeration that indicates his or her position within a *mass*. . . . In the societies of control, on the other hand, what is important is no longer either a signature or a number, but a code: the code is a *password*, while on the other hand the disciplinary societies are regulated by *watchwords* (as much from the point of view of integration as from that of resistance). The numerical language of control is made of codes that mark access to information, or reject it. We no longer find ourselves dealing with the mass/individual pair. Individuals have become *dividuals*, and masses, samples, data, markets, or *banks*" (5-6). Consequently, part of being dividual includes parceling personhood among expert discourses. As Joseph Dumit explains in his study of how medical imaging practices have done so to the brain, "medical anthropologists have long faced the relation between what Merleau-Ponty called our objective body and our lived body, or our person, with a variety of more subtle analyses. . . . These approaches understand the objective body to vary with the development (positive or negative) of technoscientific culture, attending to how the historical-cultural category of the person (via politics, economics, etc.) influences the evaluation of the objective body" (*Picturing Personhood* 157).

Unlike the individual, with which the liberal humanist subject enjoyed a unique kernel of personal subjectivity – a soul, a mole within a burrow – the dividual is a simulacrum generated from the surrounding environment, reflecting its amalgam of codes, masses, samples – a serpent extended throughout. "We have passed from one animal to the other, from the mole to the serpent, in the system under which we live, but also in our manner of living and in our relations with others. The disciplinary man was a discontinuous producer of energy, but the man of control is undulatory, in orbit, in a continuous network" (6).[32] Deleuze struggles to produce physical examples embodying the modulatory, deformative control he envisions; "enclosures are *molds*, distinct castings, but controls are a modulation, like a self-deforming cast that will continuously change from one moment to the other, or like a sieve whose mesh will transmute from point to

32 Deleuze's dividual is consonant with Jenkin's collective intelligence embodied in the monitorial citizen, although given a negative cast.

point" (4). This concept is more suited to adaptive, feedback-controlled mechanisms familiar to engineers and computer programmers.

While the notion of continuous control may have a positive connotation, such as Engelbart's Type C activity of improving improvement, Deleuze and others view it negatively when it is not exercised critically. Deleuze refers to the progressive salary structure of corporations and institutions to embody the rhetoric of continuous control over careers, which Boltanski and Chiapello connect to the ideals of the projective city. "The modulating principle of salary according to merit has not failed to tempt national education itself. Indeed, just as the corporation replaces the factory, *perpetual training* tends to replace the school, and continuous control to replace the examination. Which is the the surest way of delivering the school over to the corporation" (5). Indeed, as Andrew Feenberg points out, self-management is now so deeply rooted in consciousness that it appears to be a natural outcome of the progress of civilization, rather than a contingent effect of societies of control and class struggles.[33] He writes that "industrialism has continued to develop on the track originally set by its capitalist origins. Its central problem is still control of the labor force which, lacking ownership or identification with the firm, has no very clear reason to favor its success. The instruments of that control, management and technological design, have rooted the system so deeply in consciousness and practice that it seems the outcome of progress as such. The fact that the system has been shaped not only by technical necessities but also by the tensions of the class struggle has been forgotten" (*Questioning Technology* 40).

33 Catherine Malabou makes this seeming naturalness of dividual self-management a central theme in *What Should We Do With Our Brain?*, comparing it to the critique of Darwinism in neurobiology.

Boltanski and Chiapello urge contemporary critics to focus on the shift in justification of capitalist endeavor, and away from class struggle. The revival of capitalism following the crisis of the late 1960s led to construction of a new normative fulcrum they call the projective city. Their claim is that contemporary analyses are reformist but not revolutionary, for capitalism is very successful at assimilating critique (xiv).[34] To understand how capitalism has changed from 1965 to 1990, they follow Latour's analyses to demonstrate social dimensions of technological change. In particular, they investigate the social effects of network architecture based on the work of Emile Durkheim, James Beniger, and Fernand Braudel emphasizing social conflicts provoking technological development. Their goal is to uncover the societal projects “aiming to make the network a normative model” (xxii).

In France, the bourgeoisie had been supported by inheritance income in addition to a salary, which became the primary source, so that wage earning class of cadres – literally 'career plans' – could live the bourgeois life. However, the point as been reached that cadres and their education-based plans are losing their advantage. The flexibility in OECD countries whittling down social security via temporary workforces, flexible hours, reduced benefits, and outsourced management has led to their position that world capitalism is healthy, while societies are in poor shape, situating populations of declining human intelligence, which will be called social regression (xxxviii). Families became more fluid and fragile, transferring the social task of reproduction to schools, compounding insecurity. The paths to peak full employment and a democratized republican school system are now diminishing under the connexionist metanarrative. Yet there is no substitute for belief in progress, which has sustained the middle

34 The authors claim that the practical implications of their study are limited to the book's twelve page Postscript.

class. Multinational firms embody winners opposed to which the majority of individuals are losers merely riding technological waves not steering it as a cyberspace presence. Fatalism is the corollary of waning critique regardless of its disposition in larger scale future trends. The dividual is the post-postmodern manifestation of regressive subjectivity.

The current situation is ideological disarray; critical thought cannot keep up. Rather than trying to explain it away, their study observed variations in the spirit of capitalism, which is taken as a given. “The spirit of capitalism is precisely the set of beliefs associated with the capitalist order that helps to justify this order and, by legitimating them, to sustain the forms of action and predispositions compatible with it. . . . *Our intention is to study observed variations, not to offer an exhaustive description of all the constituents of the spirit of capitalism*” (10). They do so by articulating six logics of justification of social arrangements, articulated as 'cities' with privileged forms of expression by their great men: inspirational, domestic, reputational, civic, commercial, and industrial. The first spirit of capitalism was rooted in a compromise between domestic and commercial, the second between industrial and civic cities, and so on. The new, seventh city is modeled on 1990s management texts for cadres and concrete proposals for improving French social justice. They call it the *projective city*. “In particular, this explains why we have examined so closely the mechanisms that aim to introduce new forms of security and justice into a universe where flexibility, mobility and network forms of organization had become basic reference points for mechanisms proposed by jurists or economists, among others, which were being discussed in the second half of the 1990s. At a theoretical level, analysis of these mechanisms allowed us to give substance to the projective city—a new normative fulcrum that we think is in the process of being formed while from a more practical standpoint, it enabled us to identify some of the points which critique seemed best placed to latch on to" (xv).

Importantly, the new spirit of capitalism was derived from both theory and empirical analysis of management literature addressed to cadres for the state of the art in running firms and managing humans. It was found that corporate profit was not inspiring in either period to cadres or the general workforce, who wanted genuine reasons for engaged commitment. The distinctive problems of 1960s management literature were dissatisfaction of cadres and managerial problems of giant firms. Professionalism of management becomes its target as added layers of bureaucratic hierarchy bought about fears that the capitalist business seemed similar to collectivized and fascist firms. Solutions were offered by decentralization, meritocracy, and especially management by objectives, which also furnishes criteria for measuring performance. Consequently, foils of 1960s management literature pertain to the logic of the domestic world in favor of results-based impersonal judgment, delegitimation of traditional employers implicit in legitimation of cadres, and transition from patrimonial bourgeoisie to the bourgeoisie of managers.

In contrast to these 1960s trends, they found the distinctive problems of 1990s management literature were hierarchy, morality of domination, rigid planning, rapid technological change, re-engineering to align with the network model, and the blurring boundaries of the firm itself.[35] However, in this body of literature they also find an unscrutinized obsession with adaptation and flexibility. The cadres of the prior period are obsolete, replaced with the generalist manager, who is also distinguished from the engineer. Indeed, the manager is

35 The blurred boundaries form a major component of Spinuzzi's study of networked businesses.

the quintessential network man; other important actors in the post-postmodern firm are coaches and experts (78-79). What all these actors have in common, in conformance with the network model, is the elevation of continuously executing projects over other ideals. Projects transgress all boundaries, feeding neo-management beliefs in individualism and personal development. People are rewarded on their ability to work on a project, valuing interpersonal relations, flexibility and adaptability; hierarchical careers are replaced with succession of projects deployed to develop personal skills. One's employability is the capacity required to be called upon for projects.

Thus the project form of social organization is the new apparatus of justification. Projects delineate spaces of mini-calculation within otherwise indeterminate networks. In the projective city people are encouraged to forge links but only respect maxims specific to projects. The common, superior principle in the projective city is sheer activity, generating or integrating oneself into projects and networks, putting an end to isolation. Lesser values include autonomization and valuing the art of mediating and making connections. In this model, life is seen as a succession of projects, with the aim of extending networks by multiplying connections and proliferating links. The great man knows how to engage in a project enthusiastically, and is adaptable and flexible, thus employable.

Flexibility and adaptability derive from autonomy, not from obedience, with an intuitive talent for knowing how to select and plunder ideas. Project people must be able to interest others, be at ease, and be local. The great are integrators, enhancers of life; they are project heads, managers and coaches, in contrast to cadres. Metaphors for this rhizomorphous form include weaving, fluid flow, and biology of the brain.[36] In contrast to the flexibility and interpersonal savvy of the great man of neomanagement, the little people, i.e., the losers in the projective city, are rigid, cannot be engaged or employable on a project, and are incapable of changing projects. They may be rigid because of attachment to a single project or place, preferring security at the expense of autonomy.

The dark side of this transition in the workplace, and how it connects to the articulation of the control society dividual as the post-postmodern as-is situation, is the totalizing infiltration of the projective spirit into every aspect of social and personal life. According to Boltanski and Chiapello, the great man is a Deleuzean nomad, sacrificing impediments to availability, abandoning disinterested friendship, and is streamlined and ambivalent about moralizing. He sacrifices personality to being connexionist, being chameleon; the quiddity of self enterprise derives from his constellation of established connections (124). This is an aspect of the dumbest generation that Bauerlein misses, but just as important as the fact that youth are encased in a social cocoon of peer influence.

Far beyond Foucault's observation that hospitals, schools, and workplaces resemble prisons, and vice-versa, now family dynamics, exercise, and recreation resemble the always engaged, always planning, scheduling, evaluating, and networking habits of the project-oriented workplace. Subjectivity in connexionist networks is an effect of constitutive links. Existence is a relational attribute; disaffiliation its sole sanction. Boltanksi and Chiapello explain that “another risk, of a very new type, generated by flexible organizations is that it is much easier for actors in

36 Hayles frequently invokes Malabou regarding this crossing of metaphors between the logic of the projective city and the biology of the brain.

firms to be 'out for themselves', as popular language has it to pursue their own interests, without taking into consideration those without whom their action would not have been crowned with success" (94). Moreover, "the introduction of the 'coach', functioning as a psychologist in the firm's service, although charged with helping people to flourish, can be experienced by some as a danger of the firm encroaching on their private lives" (94). Thus they contend that development of a business ethics management discipline is related to anxiety that neo-management mechanisms be used ethically.

Life skills are elevated over knowledge and *savoir faire*, facilitating an instrumentalization of human beings in their most specifically human dimensions: "more generally, in stressing versatility, job flexibility and the ability to learn and adapt to new duties, rather than possession of an occupation and established qualifications, but also the capacity for engagement, communication and relational qualities" (98). If the postmoderns railed against regressive subjectivity for replacing the liberal, humanist core of identity with trifles, critics of the dividual find fault with the transformation of substantial subjectivity into epiphenomena of transitory network linkages. Clay Spinuzzi excellently summarizes the situation in *Network: Theorizing Knowledge Work in Telecommunications*, his ethnography of network dividuals. "The assumption that people generally make about learning and training is that individuals progressively accumulate skills; it's *developmental*. But in a network, it's far from clear that developmental models obtain: the individual herself is called into question, becoming what Deleuze calls a dividual (1995) whose work activities are fragmented and interleaved, whose attention is continuously partial (McCarthy & Boyd, 2005; Stone, 2006), whose work takes place in many organizational structures simultaneously (Castells, 1996; Drucker, 2003), whose activities are increasingly infiltrated with . . . new types of working and learning of living (Johnson-Eilola, 2005, p. 31), and whose learning is consequently more self-directed (Senge, 1994) and continual at all levels (Castells, 1996). Under such conditions, associative complexes, which Vygotsky characterized as childish (1962), become the primary way of understanding and learning about work (Johnson-Eilola, 1998, 2005)” (176).

To round out the chapter, and this introductory literature review, let me now articulate the dividual *cyborg* to complete the characterization of the as-is situation upon which the theoretical framework and methodology of the third chapter will be deployed. *Cyborg* is a contrived term implying cybernetic organism, so it can include genetically engineered biological entities, but is most often equated with technologically enhanced human beings. Per Donna Haraway, “a cyborg is a cybernetic organism, a hybrid of machine and organism, a creature of social reality as well as a creature of fiction,” of which “Michael Foucault's biopolitics is a flaccid premonition of cyborg politics, a very open field” (*Simians Cyborgs Women* 149). As a representation of boundary crossings, she points out that the distinction between humans and animals, organisms and machines being thoroughly crossed by the end of the twentieth century, so that cyborg identity marks the current confrontation between physical and non-physical, where “cyborgs are ether, quintessence” (153). Her rejoinder, which seems quaint in 2015 for its 1990s perspective, is “a cyborg world might be about lived social and bodily realities in which people are not afraid of their joint kinship with animals and machines, not afraid of permanently partial identities and contradictory standpoints” (154). Haraway urges us to consider the positive consequences of viewing cyborgs as other than enemies: “our bodies, ourselves; bodies are maps of power and identity. Cyborgs are no exception. A cyborg body is not innocent; it was not born in a garden; it does not seek unitary identity and so generate antagonistic dualisms without end. . . . Intense pleasure in skill, machine skill, ceases to be a sin, but an aspect of embodiment. The machine is not an *it* to be animated, worshiped, and dominated” (180).

As Bauerlein, Turkle, Hayles and others I have quoted argue, the contested field of interest today is over cyborg politics – how joint kinships of humans, animals, and machines comport themselves together. Haraway associates the cyborg with feminism, and her invocation of the American Southwest Indian figure of the Coyote or Trickster is appropriate for “acknowledging the agency of the world in knowledge makes room for some unsettling possibilities, including a sense of the world's independent sense of humor” (199).[37] For Hayles, the cyborg model is reached by taking the speculative leap of considering “the human and the digital computer as partners in a dynamic heterarchy bound together by intermediating dynamics. . . . Citizens in technologically developed societies, and young people in particular, are literally being reengineered through their interactions with computational devices” (*Electronic Literature* 47).

To Turkle, cyborg being has two futures: fully networked life, and evolution in robotics. In her recent book *Alone Together*, she examines how “we bend to the inanimate with new solicitude” (xii). She dubs the present time as “the robotic moment. This does not mean that companionate robots are common among us; it refers to our state of emotional and I would say philosophical readiness. I find people willing to seriously consider robots not only as pets but as potential friends, confidants, and even romantic partners” (9). In her critical evaluation of

37 The Trickster figure is frequently invoked as the ironic stance computer hackers sometimes deploy to justify their activities that have been deemed illicit.

posthuman cyborg identity, she sees the computer as having gone far beyond the second self, the mirror of the mind that launched her career in the early 1980s as an ethnographer of computer cultures. “Our new devices provide space for the emergence of a new state of the self, itself, split between the screen and the physical real, wired into existence through technology” (16). Roboticists have learned triggers that help us fool ourselves, perhaps making us more gullible, or more striated in our range of emotional and cognitive reactions to them. Her great fear is that the contemporary response to robots infiltrates and realigns our response to ourselves, other humans, and animals. It is in this sense that the dividual paradigm envelops the cyborg.

Revisited from the perspectives of the 1970s and 1980s, premonitions of the cyborg future glimpsed in popular culture seem radical in comparison to the unreflective complacency to which Turkle alludes with the robotic moment. Baudrillard philosophizes from the perspective of an already immanent precession of simulacra having taken over nature, institutions, and the human spirit, but also from the outside. Thus his terse, negative bias toward computing technology. “Programmed microcosm, where nothing can be left to chance. Trajectory, energy, calculation, physiology, psychology, environment nothing can be left to contingencies, this is the total universe of the norm—the Law no longer exists, it is the operational immanence of every detail that is law” (*Simulacra and Simulation* 34).[38] Under this rubric, in the dividual fashion, procreation occurs as the joint act of cloner and clone in service of the matrix. “The cloner does not beget himself: he sprouts from each of his segments. . . . The Father and the Mother have

38 Baudrillard's putatively critical position neglects awareness of the messiness and materiality of software that SCOT theorists that will be introduced in chapter three articulate, which in turn forces reflection back on the putatively totalizing logic of programmability.

disappeared, not in the service of an aleatory liberty of the subject, but in the service of a *matrix called code*" (96). The programmed microcosm engenders a programmed macrocosm, leading to docility through socialization and political reactions that are implosive instead of explosive. "There lies the true nuclear fallout: the meticulous operation of technology serves as a model for the meticulous operation of the social. Here as well, *nothing will be left to chance*, moreover this is the essence of socialization, which began centuries ago, but which has now entered its accelerated phase, toward a limit that one believed would be explosive (revolution), but which for the moment is translated by an inverse, *implosive*, irreversible process: the generalized deterrence of chance . . . doomed to the descriptive transparency of mechanisms of information" (34-35). Deterrence of chance becomes the overriding global police action. Latent in Baudrillard's postmodern musings are the plots of countless science fiction literary works and films from the period, as well as startling similarities to the global police actions of the "War on Terror."

Paul N. Edwards layers the Cold War military industrial complex over the familiar postmodern backdrop, providing ample historical evidence and examples from popular culture of a *closed world* mindset in both the United States and the Soviet Union. The metaphor derives from the feedback control model of a dome of global oversight the was sought by the two superpowers (*Closed World* 1). In his narrative, human control over autonomous technology paves the way to the robotic moment. Techniques, technologies, practices, fictions, and languages formed closed world discourse. Cyborg discourse of human automata took a particular trajectory through creation of iconographies and political subject positions that persisted into the 1980s in the United States, one that aligns with conventional cultural stereotypes (27). For example, the movie *Terminator 2* “seems to suggest that we are waking up from a nightmare run by cyborg Others to a world in which we control them. . . . The figures of the violent, unfeeling, but rational and devoted father-protector and the emotional, unstable, but loving mother return to form the basis of the post-Cold War social order. The cyborg-machine returns to its place as technological servant, intimating a renewed sense of human control over autonomous technology” (362).

Hopes for liberatory cyborg politics dissolve into disingenuous multiculturalism, leading to an easy relativism, with nothing besides the postmodern embrace of cultural chaos to fill voids left by prior great narratives. We get retrenchment in nationalism, fundamentalism, and global trade. “This simplistic doctrine flattens all cultural difference into two categories. Exotic differences function as the interesting resources for a Believe-It-Or-Not pseudo-anthropology. . . . It is all too easily transformed into the utterly mundane, and anthropology gives way to Disney World” (363). As Edwards closes out his study, he reaches the conclusion, shared by Haraway and others, that the only response to the closed world is to radically exercise the degrees of freedom its boundaries offer, becoming hybrids, even monsters. “*Recombination* to appropriate a 1980s biotechnological information metaphor is the only effective possibility for rebellion in the closed world: taking the practices of disassembly, simulation, and engineering into one's own hands. Coming to see oneself as a cyborg, fitting oneself into the interstices of the closed world, one might open a kind of marginal position as a constructed, narrated, fragmented subjectivity, capable of constant breakdown and reassembly. . . . Subverting the closed world by an interstitial engagement rather than a green-world transcendence, the cyborg becomes an always partial, self-transforming outlaw/trickster living on the margins of panoptic power by crisscrossing the borders of cyberspace" (341-342).[39]

Whereas Edwards argues recombinant cyborg subjectivity to be the only rebellion in a closed world, I will further complicate the analysis of the as-is situation by reminding us of our postmodern heritage. As central Florida has been hailed the kitsch capital of the world, the dividual lives in Disney. Dividuals, as second-order amalgams of simulacra – concretizations of contingent institutional and commercial pilings – can neither seek return to an untarnished, natural core, which Edwards calls the green world, nor recast themselves entirely as baseless avatars, self-made virtual beings. Nicholas Gane urges that we remember how our cyborg subjectivity is situated within the built environment that seeks to optimize itself in terms of performativity ("Computerized Capitalism" 436). As a result, "in our day-to-day processing of short 'bytes' of information we ourselves become more like machines. In other words, through our use of new media technologies, we, as humans, become increasingly 'inhuman'" (441). He homes in on Lyotard's contention that cultural processes transformed through the affordances of inhuman practices do not further liberate the spirit by loosening the biological constraints on communication, but instead dumbs them down; "the digitalization of data tears both cultural artifacts and sensory experience from their moorings in physical time and space" (442).

39 Baudrillard gives this discourse what Boltanksi and Chiapello will explain is an 'artistic critical' cast, whereas Edwards' closed world is decidedly 'political' in their terms. Baudrillard's exotechnical/esotechnical division acknowledges consummation of Nietzschean metaphysics, yielding a default mode of real virtuality production, an assault on subjectivity that is built into media with which our identities co-constitute as embodied consciousness such that media are no longer separable soul equipment but focal to authentic being.

Gane reasons that “the most effective streams of cultural capital are those that can be transmitted, received and decoded (consumed) with the greatest ease. In these ways, the inhuman (technologized time) has invaded both the human (life itself) and culture, and in the process has strengthened the grip of the capitalist market. . . . In other words, even the most radical forms of thought and identity, including those at the heart of the 'postmodern condition', have been captured by the capitalist system and commodified” (445-446). No better example than the dumbest generation outlined by Bauerlein shows how this recombinant spirit has been appropriated by the global capitalist market to equip young souls with apps, memes, and data trails for their lifetime traversal of the technological era. Besides a few coyote trickster hackers, the general destiny is not toward recombinant cyborgs but toward *recumbent* ones – toward the passengers of the *Axiom* spaceship from *WALL-E*, floating on their self propelled recliners immersed in virtual realities that are little more than recreations of humans from our era watching television commercials.

CHAPTER 3: THEORETICAL FRAMEWORK AND METHODOLOGY

As I proposed in the introduction that we pay attention to shifting bases of cognition noted by Manovich, from foregrounding the symbolic, textual, discursive aspect to embracing the concurrent visual, haptic, embodied components, a sea change in our theorizing the technologized lifeworld ought to be noted as well. The key to transforming the regressive subjectivity that has yielded the dumbest generation, I will argue in developing my theoretical framework and methodology in this chapter, is to shift the underlying epistemological and metaphysical basis from what I am calling *Foucault's dust* to Pierre Lévy's *collective intelligence*. Foucault makes this analogy in *Discipline and Punish* with regard to the development of bureaucratic organization. “Police power must bear 'over everything': it is not however the totality of the state nor of the kingdom as visible and invisible body of the monarch; it is the dust of events, actions, behavior, opinions—'everything that happens'. . . . And, in order to be exercised, this power had to be given the instrument of permanent, exhaustive, omnipresent surveillance. . . . And this unceasing observation had to be accumulated in a series of reports and registers; throughout the eighteenth century, an immense police text increasingly covered society by means of a complex documentary organization” (213-214).

Knowledge gathers like dust, accumulates sedimentary layers from which relatively stable, deterministic progressions are retrospectively discerned and documented. Likewise, artifacts concretize their past histories of innovations, adaptations, even advertisements and promotional rebrandings. This ontological stance is adequate to describe a modernist world devoid of divine influence, constituted through the necessities of physical laws and statistical averages. Its boundary of sense just barely accommodates postmodern precession of simulacra, which insist that the organizational principles of the physical world be extended to embrace the idiosyncratic characteristics of highly engineered, virtual worlds obeying their own internal logic.

Hardt and Negri refer to the appeal of conspiracy theories to explain this epistemological condition that Latour credits to the constitutional guarantees of the moderns, where nature and the social are constructed but seem natural, distinct from mediation under the crossed out God (*We Have Never Been Modern* 32). They write: “as Fredric Jameson explains wonderfully in the context of contemporary film, conspiracy theories are a crude but effective mechanism for approximating the functioning of the totality. The spectacle of politics functions *as if* the media, the military, the government, the transnational corporations, the global financial institutions, and so forth were all consciously and explicitly directed by a single power even though in reality they are not" (323). Perhaps the prior analysis of regressive subjectivity should be amended with the suggestion that it results from default comportment to the technological Big Other in which information accrues like the settling dust of everything that happens, yet the patterns that emerge seem to match our expectations as by conspiring agents.

Recall McLuhan's appeal to the Schmoo fantasy to characterize everyday belief about how commodities are produced: once a design is imagined, the magical powers of industrial capitalism manufacture it perfectly to specifications. Consonant with postmodern cynicism over loss of authenticity to the iron rule of commerce, we experience the dual action of taking wish fulfillment for granted while believing in free will as far as the authenticity of desires and preference formation is concerned. However, our guiding metaphysic ought to account for the influence of not only such programmed visions, which yield the appearance of global conspiracy even when there is no conscious agent at the core, but also emergent phenomena from the machinic phylum as well, the unnamed high commands of cyberspace beyond our ken that have been brought into being as the global Internet and now, in the manner Kittler paraphrases Lacan, never cease to write themselves.

Theoretical Framework

The theoretical framework I will present in this chapter combines critical theory, textuality studies, and media studies, with approaches based on the social construction of technology (SCOT) applied to scholarship on the history of computers, software, and networking.[40] These are complemented by psycho-social studies of computer programmers, ethnographies of programming communities, and positions by philosophers of computing technologies. Arranging the theoretical framework continues by extending these perspectives with the suggestions of subdisciplines that have emerged in the last decade in the digital humanities: software studies, game studies, critical code studies, recent attention to code space, and platform studies. Bogost, who argues that playing games teaches the underlying procedural structures of everyday activities as effectively as through programming them, provides the key concept of *procedural rhetoric*. With Montfort he offers up a methodological framework in their explication of platform studies that accommodates multiple, concurrent analytical levels applied to the same phenomena, laying out the previous perspectives in a model with a curious resemblance to the OSI and TCP/IP network layers. Table 1 provides a summary view.

40 These are also the primary areas covered by my coursework in the Texts and Technology program, whereas I delved into the latter specialized components during my candidacy exam and dissertation research.

Table 1: Summary of Theoretical Framework

Section	Key Theorists	Key Ideas
Critical Theory	Iser	Phenomenology and hermeneutics
	Kellner	Multiperspectival social theory
	Boltanski and Chiapello	New spirit of capitalism; textual analysis
Textuality Studies	Barthes	Second-order semiological systems
	Buzzetti and McGann	OHCO thesis; poiesis-as-theory
Media Studies	Kittler	Convergence; time-axis manipulation
Social Construction of Technology	Latour	Quasi-objects
	Bijiker	Multidirectional model
	Callon	Engineer-sociologists; actor networks
Histories of Computer, Software, Networking	Aloisio	Etymology of computer
	Aspray and Campbell-Kelly	Importance of industry periodicals
	Edwards	Closed world discourse
	Shasha and Lazere, Lammers	Interviews and biopics
	Hafner and Lyon, Abbate	Contested narratives
Psycho-Social Studies of Programmers	Brooks, Jr.	Vicissitudes of software engineering
	Weinberg	Reading code, egoless programming
	Kraft	Structured programming as deskilling
	Turkle	Hard and soft mastery programming styles
Philosophers of Computing Technologies	Weizenbaum	Fetishization of computer power
	Golumbia	Computationalism; presentism
	Galloway	Material immanence of protocol
Software Studies	Bogost	Unit-level analysis
	Fuller	Critical, social and speculative software
	Chun	Programmed visions; sourcery
	Kitchin and Dodge	Code/space
	Berry	Unreadiness-to-hand; parasitic subjectivity
Code Studies	Marino	Critical code studies
	Chun, Fuller, Berry	Phenomenology and ontology of code
	Tanaka-Ishii	Programming languages as semiotic probes
	Montfort et. al.	Critical reading of single lines of code
Procedural Rhetorics of Diachrony in Synchrony	Kirschenbaum	Extreme inscription
	Kittler	Protected mode; writing under Microsoft
	Monfort and Bogost	Platform studies
	Bogost	Procedural rhetoric; procedural enthymeme
	O'Gorman	Diachrony-within-synchrony

Multiple, conjoined explanatory layers operating in different temporal orders of magnitude provide an ontology appropriate to decentralized, distributed networks of dividual cyborgs. I call the combined methodological framework for developing a philosophy of computing the *procedural rhetoric of diachrony in synchrony*. Applying this expansive theoretical framework and following the lead of other thinkers, I present a two-pronged methodology, the first part calling for investigation of *philosophical programmers* summarily sketched for future research, in order to focus on the second, *working code places*, which offer *programming philosophers* the novel method of *critical programming* that has been the practice I have cultivated through the development of a number of software projects, including the one serving as my machinic companion throughout my candidacy exams and the creation of this dissertation.

Critical Theory

According to Wolfgang Iser, hard-core theory predicts, developing laws, as in the sciences, whereas soft theory maps, developing metaphors, as in the humanities (*How To Do Theory* 5-6). Methods, on the other hand, provide tools for interpretive processes. “Hence there are two types of theory in the humanities: those that have to be transformed into a method in order to function, and those that are applied directly, retroactively undergoing a diffraction of their categories" (11). As humanities theories emerge to address questions concerning our comportment to technology, and computing machinery in particular, many instances of both types of theory are often mixed together to produce novel, hybrid analyses, feeding back the results of methodological tools and causing the diffraction of theories from related disciplines.

Moreover, embodiment and context are always relevant to the work of art; “in contradistinction to aesthetics, then, theories of art derive their components from sources outside themselves, thus obtaining a more reliable basis than the contrived speculations of aesthetics could ever provide” (9). In a literal sense technological artifacts are also works of art, as manufactured objects in contrast to naturally occurring phenomena, so they are likewise wrapped in humanity by their designers and end users. Thus, the beginning of a theoretical framework for addressing the human-machine symbiosis of our times could be informed by the various theories of art Iser presents: phenomenological, hermeneutical, gestalt, reception, semiotic, psychoanalytic, Marxist, deconstruction, generative anthropology, pragmatism, feminism, and finally postcolonial discourse.

Phenomenological theory “conceives the work of art as an intentional object to be distinguished from real and ideal objects,” and focuses on intentional acts to gain insight on ways we relate to the world (163). A key concept Iser presents is concretization, as developed by Roman Ingarden, who theorized about literary works of art. “The text is given as a layered structure through which the subject matter of the work can come to light, but the actual bringing to light occurs in an act of *concretization*. Thus the literary work has two poles, which we might call the artistic and the aesthetic: the artistic refers to the text created by the author, and the aesthetic to the realization accomplished by the reader. . . . The work is the point of convergence, since it is located neither in the author's psyche nor in the reader's experience. . . . Hence, according to Ingarden, it is an intentional object, whose component parts function as instructions, the execution of which will bring the work to fruition” (14-15). Transferring this model to software, the artistic pole may be seen as the source code created by programmers, and the aesthetic as the realization accomplished by its incorporation and execution in sundry computing systems. Indeed, philosophers of technology like Gilbert Simondon and Andrew Feenberg use concretization to describe the layering of engineering designs in successive generations of an artifact.

Hermeneutics, a second theoretical basis Iser presents as a means of understanding “how the gaps between text and recipient as well as between past and present were to be negotiated,” (29) also fits well for the study of cyborg identity. Gadamer's concept of tradition can be applied to the brief history of electronic computing because many now view it as a set of local traditions, as the various, often competing narratives about the development of early electronic computers, networking, and the personal computer attest. I argue these traditions should be covered along with other cultural and historical studies to counteract the prejudices of digital natives who do not even recognize a time before pervasive computing. As I suggested in the first chapter, von Neumann invites us to resuscitate the Socratic maxim to know thyself by examining our machines, too. As Iser put is, “thus otherness turns into a mirror for self-observation and such a relationship sets the process of self-understanding in motion because the alien that is to be grasped realizes itself to the extent to which one's own dispositions come under scrutiny” (36). However, he notes that “resuscitating the past for the purpose of self-understanding requires a methodologically organized approach, because otherwise individual arbitrariness would prevail” (37-38).

R.G. Collingwood's question and answer logic is one such method derived from hermeneutic theory that strongly resembles reverse engineering, which is often used in the analysis of technological systems, especially software. “Each work of art is to be conceived as an answer to a question or problem prevalent in the respective historical situation within which it was produced. The work as an answer is bound to contain the question in the form of an issue that had to be addressed. Through the logic of question and answer we are able to reconstruct the context of the work to which it has reacted, thereby making us present to a historical situation that has never been our own. Thus a truly historical interpretation of the work of art emerges, which allows us both to reenact the work on its own terms, and to begin to understand its otherness" (39). Hayles' research into the Macy Conferences on cybernetics reflects this hermeneutic method as she teases out the various contested narratives that were in play before what became the tradition concretized around a particular set of texts, models, and theorists.

A similar exercise could be conducted for the other theories of art Iser surveys. However, I want to get to *critical* theories that imply moral judgments about their subjects embedded in their interpretations. The Frankfurt school, notably Horkheimer and Adorno, are often cited. According to Douglas Kellner, "they developed the first left critiques of the mass society and provided early warnings concerning the decline of individuality and freedom and threats to democracy in the brave new world of consumer capitalism. . . . In retrospect, *Dialectic of Enlightenment* is an extremely interesting text in that it provides the first critical questioning of modernity, Marxism, and the Enlightenment from within the tradition of critical social theory" ("Critical Theory Today" 44; 50). Their primary concern is reason instrumentalized and incorporated into the structure of society, sinking into new barbarism. Mathematics reified thought as a machine process. "In the preemptive indentification of the thoroughly mathematized world with truth, enlightenment believes itself safe from the return of the mythical. . . . Despite its axiomatic self-limitation, it installed itself as necessary and objective: mathematics made thought into a thing—a tool, to use its own term" (*Dialectic of Enlightenment* 18-19).

The comparison to Hayles' analysis of how we became posthuman is inescapable. They also provide criticism of the culture industry that is often applied to movies and video games today. Kellner purports that "what is new, however, is that the irreconcilable elements of culture, art, and amusement have been subjected equally to the concept of purpose and thus brought under a single false denominator: the totality of the culture industry. . . . With good reason the interest of countless consumers is focused on the technology, not on the rigidly repeated, threadbare and half-abandoned content" (107-108).

Kellner argues that revisiting such classics of the Frankfurt school helps contemporary, interdisciplinary social theory address the new stage of state and monopoly capitalism. Both regressive subjectivity and the new spirit of capitalism were covered in the previous chapter. What is special about the *Dialectic of Enlightenment*, Kellner contends, involves Horkheimer and Adorno's methodology, using "the techniques of philosophical and literary interpretation to unfold the social truth contained in literary and philosophical texts. This move decenters the sort of analytic social theory that constituted the critical theory of the 1930s and marks a significant departure and growing mistrust of social sciences and theory" (52). Recall the discussion of Odysseus as the trapped office worker, frustrated by his inability to transcend the confines of the bourgeois role.[41] This and other examples of what Kellner calls multiperspectival social theory take inspiration from Nietzsche. "The experimental form of the *Theses on Anti-Semitism* thus provides a multiperspectival model for social theory that is still useful for social theory today.

41 Kellner provides examples of recent Frankfurt school scholarship by Wiggershaus and Habermas.

Multiperspectival theory is open, non-essentialist, and cultivates new ways of seeing and thinking" (52)

Kellner has a bone to pick with other postmodern critics. "Baudrillard, Lyotard, and other forms of postmodern theory which reject macrotheory and political economy, or wallow in implosion, fragmentation, irony, and nihilism, lack the theoretical resources to develop a critical theory of contemporary society. Instead, theories are needed that articulate both fragmentation and new forms of social structuration, that stress disorganization and reorganization within the present order, and that combine macro and micro perspectives" (58). Hayles, on the other hand, consciously duplicates Adorno, Benjamin, Marcuse by engaging aesthetic works of science fiction to elicit social truths. Moreover, she answers the challenge of articulating fragmentation and new forms of social structuration, macro and micro levels, by utilizing methods developed by Frankfurt School theorists.

Plenty of thinkers have commented on the impacts of automation on global productivity, gesturing to this transformation of the inhuman, and there are numerous critiques of globalization, yet according to Boltanski and Chiapello, theory has stagnated in establishing mechanisms to control the new forms of capitalism (xvi). They construct a framework for combining critical and pragmatic sociology based on the analytic framework of a prior work in which Boltanski collaborated, *De la justification*, affording analysis of supra-individual entities, focusing on 1965 through 1995 (xii). They were inspired by perplexity over coexistence of declining social positions of the masses—the vanishing middle class—and booming capitalist economies witnessed in the last two decades of the twentieth century, claiming that "a *critical sociology* indifferent to the values that actors claim to adhere to must therefore be replaced by a *sociology of critique*” (xi). In particular, they make the distinction between social and artistic critique among “the bearers of these various grounds for indignation and normative fulcra have been different groups of actors, although they can often be found associated in a particular historical conjuncture” (38). Artistic critique foregrounds loss of meaning, loss of a sense of the beautiful, such as Baudelaire's bourgeoisie and the dandy exemplifying attachment and detachment, whereas social critique emphasizes exploitation and dehumanization as its targets.[42]

Traditional political philosophy has not yet attempted to justify the network, connexionist order. Critique seemed to miss the advance of new network mechanisms of capitalism besides condemning exclusion, until recently, though its 1970s vanguard emerge as promoters of the transformation (156). Capitalism itself readily submits to what they call the exit critique, the point at which consumers decide not to buy, employers not to hire, or providers to not serve.[43] Where they have been attentive, historicist and naturalist efforts compete to construct scientific sociologies based on networks: “the tension between a historicist position (the network is the form that suits our age) and a naturalistic position (the network is the texture constitutive of any social world, even of nature in its entirety) can be reduced if one accepts that in the order of knowledge, reticular organization constitutes the form that is best adjusted to the global vision of the world from the viewpoint of a city founded upon a connexionist logic” (151).

The underlying methodology of *The New Spirit of Capitalism* involves analysis of management texts published for use in business schools from the 1960s through the 1990s. Cadres and engineers are the primary recipients of management discourse. “We shall see how

42 Consider Lanier's *Who Owns the Future?* as an example of social critique arising from technologists.

43 The exit critique also motivates software pirating and development of free, open source software.

management discourse, which aims to be formal and historical, general and local, which mixes general precepts with paradigmatic examples, today constitutes the form par excellence in which the spirit of capitalism is incorporated and received" (14). Their comparative method places emphasis on differences between the two corpora. The authors conducted their textual study with two phases of analysis based on two sets of sixty texts per period: close reading by humans to define hypothetical characteristics of each period, and machine reading via `Prospero@` analytical software to corroborate them. "We used a textual analysis software program to compare the two corpora systematically. In Appendix 3, readers will find a presentation of this work, offering statistical confirmation of the interpretation of their content we have just presented. . . . Let us reiterate that, in order to meet the constraints of the test to which we are subjecting them, these texts must present engagement in reformation as a personally exciting venture, demonstrate that the measures proposed are justifiable in terms of the common good, and, finally, explain how they will deliver to those who invest in them a certain form of security for themselves and their children" (86). The results of their analysis yielded the projective city that was discussed in the previous chapter in the context of the dividual cyborg.

Mapping grammars of seven worlds via word categories shows dominance of industrial logic in both eras, and network logic overtaking domestic logic for second place in 1990s. Textual analysis brings network logic to the top position: "the program of textual analysis that we have employed thus makes it possible to bring out a major transformation in the space of thirty years in the registers of justification on which management literature bases itself, and an increase in the popularity of the network logic to top position. . . . Hence this tends to confirm the hypothesis that the construction we have extracted from texts does indeed represent, in stylized and concentrated form, what characterizes the new spirit of capitalism in a highly original fashion" (137). Communication, complexity, and chaos are predominant terms. Importantly, few management texts in this representative corpus reference authors from human sciences and philosophy; they mostly reference each other. "However, although a large number of the terms or notions drawn from management texts where network logic predominates have their equivalent in writings from the human sciences, direct references to these works are rather rare in our corpus, and pretty much concentrated under the signatures of a few authors. These authors associate management in network form with three terms: first, communication (represented by references to Habermas, Bateson, and Watzlawick); secondly, complexity (J.-P. Dupuy, Edgar Morin); and, finally, disorder, chaos and self-organization (represented by references to Prigogine, Stengers, Atlan, Heisenberg, Hofstadter and Varela). As a general rule, the authors of our corpus predominantly cite other management authors, and frequently one another; this accords with the existence of management as a specific discipline" (139). They note traces of 1970s Ivan Illich, although rarely cited by management authors. "Their anti-authoritarian emphasis, critique of centralization, stress on autonomy and on what might, with a certain anachronism, be called self-organization, and also their technological humanism placing tools in the service of humanity, not vice versa were to be taken up in the thematic of the projective city" (139). Thus it can be argued that the new spirit of capitalism arose from within its own ranks, and a sort of default philosophy prevails, sustained by its own internal feedback.

A similar study further investigating the default philosophy of computing that I touched on in the first chapter could be conducted by analyzing texts used in computer science and information technology education, in addition to the writings of philosophically minded programmers. However, the present endeavor inaugurates an entirely new form of digital humanities scholarship that engages philosophies of computing by insisting that a non-trivial percentage of intellectual labor be conducted by programming, working code, in a critical manner open to reflexive insight and iterative steering of research agenda discovered in the process of software development. In the arc of sketching a theoretical framework to respond to the as-is situation of the post-postmodern network dividual cyborg, and advance critical thought toward a philosophy of computing, textuality and media studies employed by Hayles and other digital humanists join the familiar scholarly approaches of social critics like Boltanski and Chiapello and Hardt and Negri. Following Marx, we need to go beyond interpreting, and change the world precisely at this point philosophy and computing intersect—not in the incestuous domain of cognitive science, but cultural and social history. As Jerome McGann puts it, “information scientists and systems engineers will be (already are) much involved with these changes. . . . But it is the literary scholar, the musicologist, the art historian, and so on, who have the most intimate understanding of our inherited cultural materials. Hence the importance that traditional scholars gain a theoretical grasp and, perhaps even more important, practical experience in using these new tools and languages" (*Radiant Textuality* 169).

Boltanski and Chiapello, and no doubt countless other contemporary social theorists and academic scholars, use software to analyze data and craft their texts. I contend that this practice, while valuable and widespread, is not yet *critical* programming. Instead, we should make McGann's style of textuality studies, in addition to programming itself, count as doing theory by adding more deliberate senses of working code than they do, for which the following progression refines the theoretical framework. Then we can meditate upon the nuances of its changes and their meaning through theory-as-poieisis like those of `tapoc` shaping this dissertation as its is being programmatically written through multiple iterations, for sometimes programmed visions are intentional, and well-intentioned, as programmers and systems engineers know.

Textuality Studies

Textuality studies make an obvious sweet spot and entry point for a digital humanist interested in studying computer program source code and programming languages. According to McGann, the familiar sense of text is a rhetorical sequence of page units, each with an assumed organization (96). Acknowledging that the page form is only one manifestation, Hayles defines *text* as an abstract artistic entity, “the ideal construction toward which textual editors move by collating different editions and copies to arrive at their best guess for what the artistic creation should be. . . . It is important to note that the work is ideal not in a Platonic sense, however, for it is understood to be the result of editorial assumptions that are subject to negotiation, challenge, community norms, and cultural presuppositions” (*My Mother Was a Computer* 92). To Spinuzzi, with a nod to Latour, “texts are *inscriptions* that represent phenomena, belong to *genres* that construct relatively stable relationships, and function as *boundary objects* that bridge among different activities. Texts create circulating rerepresentations: representations that themselves become represented by other representations” (148). The familiar sense of computer program code based on the von Neumann architecture implies organized rhetorical sequences, whether the consumer is the machine itself, or human technicians. These texts are abstract entities in Hayles' sense, re-representations whose ontic status is often indeterminate and made manifest on demand.

Ong describes texts as inherently contumacious, i.e., willfully and obstinately disobedient. “Writing establishes what has been called 'context-free' language or 'autonomous' discourse, discourse which cannot be directly questioned or contested as oral speech can be because written discourse has been detached from its author” (77). Neel contends that the play of absences inherent in written texts founds modern thought. “We in the West, in other words, believe we can know what we think not only without saying it or writing it down, but also without thinking it to ourselves in a linguistic way. Deconstruction attempts to show that what presents itself as thought begins in exclusion . . . *différanc*e, or the play of absences, founds thought. The Cartesian *cogitato, ergo sum* is an effect of language, not an origin or foundation" (*Plato, Derrida, and Writing* 170-171). This notion hinges on the belief that we humans think only in signs, as articulated by Saussure in proposing semiology as the science of signs, yet without appeal to a justifying external reference, according to Derrida in proposing grammatology as the science of writing. “Science of the arbitrariness of the sign, science of the immotivation of the trace, science of writing before speech and in speech, grammatology would thus cover a vast field within which linguistics would, by abstraction, delineate its own area, with the limits that Saussure prescribes to its internal system and which must be carefully reexamined in each speech/writing system in the world and history” (*Of Grammatology* 50).

The appeal of semiotics to unite disparate linguistic symbols and uncover an overarching metalanguage dissolves in the conclusion Derrida reaches that it is all *bricolage*, an ad hoc creation made from a diverse range of available pieces, more akin to Trukese sailing than European navigation, to recall the discussion of Suchman in the first chapter. “The only weakness of bricolage – but, seen as a weakness is it not irremediable? – is a total inability to justify in its own discourse. The already-there-ness of instruments and of concepts cannot be undone or reinvented. . . . The idea of the engineer breaking with all bricolage is dependent on a creationist theology” (138-139). Yet if any semiotic systems do escape this condition of interminable analysis, they are the artificial languages designed by engineers to program computers.

Textual studies are an obvious place for humanities-based philosophies of computing to develop, for human-machine interfaces have primarily evolved as skeumorphs of print. As McGann puts it, “we want to work with digital tools as we have always worked with our tools for making simulations—for instance, with all our semiotic tools, and pre-eminently with the book. Critical reflection emerges in the mirroring event that develops at simulacral interfaces, of which the book is the one we are most used to using” (214). In skeumorphs, design cues are taken from natural objects and prior means of production, such as imitation stitching molded into an automobile steering wheel, and the circular wheel itself resembling that of a sailing vessel.

Consider Roland Barthes' treatment of myth. Study of myth involves sensitivity to semiology and ideology. “It can consist of modes of writing or of representations, not only written discourse, but also photography, cinema, reporting, sport, shows, publicity, all these can serve as a support to mythical speech. . . . Mythical speech is made of a material which has *already* been worked on so as to make it suitable for communication: it is because all the materials of myth (whether pictorial or written) presuppose a signifying consciousness, that one can reason about them while discounting their substance" (“Myth Today” 80-81). Myth does not function without this prior familiarity with its base materials. Barthes calls it a *second-order semiological system* in which signifier, signified, and sign operate in a special way: “whether it deals with alphabetical or pictorial writing, myth wants to see in them only a sum of signs, a global sign, the final term of a first semiological chain. And it is precisely this final term which will become the first term of the greater system which it builds and of which it is only a part" (85). Because myths are second-order semiological systems, there is a functional equivalence of all media as constitutive language-objects.

The repetition of pattern through different forms reveals the intention of myth, Barthes suggests. The surplus apparently encoded in the signifier via, among other operations, myth, touches upon the asymptotic approach of human sign system functions to the absolute predictability and reliability of machine control operations, the architecture-specific 'machine languages' the the base of the new Tower of Babel. At the shimmering signifier boundary are hyperlinks, to which Barthes' conclusion seems well fit: "there is no regular ratio between the volume of the signified and that of the signifier. In language, this ratio is proportionate, it hardly exceeds the word, or at least the concrete unit" (90). Myth operates upon established systems, meanings become forms, concepts through signs in signification: can this second order character of myth also supply methodology to other second-order systems, such as technological artifacts, including source code artifacts and program-generated virtual reality phenomena? Kumiko Tanaka-Ishii takes up this philosophical technique of disciplinary boundary crossing in great detail in her study of the semiotics of programming languages, which will be covered later in this chapter.

Textuality studies fuel a phenomenological analysis of the programmed, machinic milieu, for the very word *program* implies the premeditatated, deliberate arrangement of writing, text, symbols to control subsequent symbolic operations as well as physical processes. Bolter points out, on the one hand, that the strict requirement of textual unity and homogeneity is relatively recent, coming "from the published work we have read, or more generally, from the current divisions of academic, literary, and scientific disciplines, which themselves both depend on and reinforce the economics of publishing" (*Writing Space* 10). This cultural bias is under assault, on the other hand, by electronic texts that "can tailor itself to each reader's needs, and the reader can make. . . . This ideal of cultural unity through a shared literary inheritance, which has received so many assaults in the 20th century, must now suffer further by the introduction of new forms of highly individualized writing and reading" (11). We are not, however, reverting to a milieu prior to print, for the mechanisms by which these highly individualized texts, which are dynamically manufactured simulacra of printed paper artifacts, are produced, belie pervasive *pro*-gramming. Bolter's suggestion that our expectations about texts derive in part from our book buying culture mirrors Hayles' idea that cybernetics evolved around the expedient scientific models of the mid twentieth century, yet both now retrospectively seem to have deterministic influence on the exemplars upon which they are analogically defined.

There is one particular pillar of the current metaphysics of textuality that is continually fortified by programming. McGann presents it while discussing the design of Standard Generalized Markup Language (SGML). "This hypergrammar treats its documentary materials as organized information, and it chooses to determine the systems of organization as a hierarchy of nested elements; or, in the now well-known formulation: text is an ordered hierarchy of content objects (the so-called OHCO thesis)" (139).[44] The topic arises repeatedly in the

44 McGann and others dismiss SGML for productive, working code, and when philosophizing, use examples from HTML and now XML, which as less complex languages, are both better suited for practical use in examples of scholarly work and at the same time incorporate more crude conversions and conventions, more 'spaghetti code',

contributed chapters of Burnard, O'Keefe and Unsworth's *Electronic Textual Editing*. Markup languages like the Text Encoding Initiative (TEI) require that all texts have underlying structures, enabling their markup. Buzzetti and McGann argue that “in principle, markup must therefore be able to make evident all implicit and virtual structural features of the text. . . . The crucial problem for digital text representation and processing lies therefore in the ability to find consistent ways of relating a markup scheme to a knowledge-representation scheme and to its data model” (62). Then they discuss the insufficiency of the OHCO thesis for missing structural mobility, assuming meaning embedded in syntactic form, and assuming coincidence between syntactic and semantic forms.[45] “A formal representation of textual information does not require an absolute coincidence between syntactic and semantic logical form. In this respect, the role of markup can be of paramount importance in bringing their interconnections to the fore” (66).

Buzzetti and McGann urge scholars to heed the existential imperative to build devices in digital space. “This necessity is what Charles Sanders Peirce would call a pragmatistic fact: it defines a kind of existential (as opposed to a categorical) imperative that scholars who wish to make these tools must recognize and implement. . . . In fact one can transform the social and

bricolage. We all know the TEI is based on XML: is this an inferior basis than something including more sublime and productive languages like C++? Legitimate philosophical debates about the comparative merits of particular programming languages beg for theorist-practitioners making claims based on long term use.

45 Other contributors to this volume who discuss the affordances and shortcomings of the TEI and other markup languages, and indirectly support or challenge the OHCO thesis, include Fraistat and Jones' article on using the TEI for encoding poetic text at level of structure, Van Hulle's musings about transclusive flexibility and the problem of overlapping hierarchies, and Huitfeldt's discussion of how the Wittgenstein manuscripts provide almost every imaginable complicating variation for textual markup and require keen awareness of the nuances of diplomatic reproduction.

documentary aspects of a book into computable code. . . . We were able to build a machine that organizes for complex study and analysis, for collation and critical comparison, the entire corpus of Rossetti's documentary materials, textual as well as pictorial” (69-70). At the same time, McGann bemoans digital humanities scholarship as missing depth for not building critical and reflective functions into the deep components of these devices. “Unlike works imagined and organized in bibliographical forms, however, these new textual environments have yet to develop operational structures that integrate their archiving and editorial mechanisms with their critical and reflective functions at the foundational level of their material form, that is, at the *digital/computational level*” (17).

McGann admits that his scholarly editing theory actively evolved while working on the *Rossetti Archive*. Traditional distinctions between critical and fascimile editions were challenged by the new possibilities of a digital text of enormous proportions. Indeed, building the Rossetti archive provided more than an analogical bridge between humanities scholarship and software engineering: it *was* software engineering. “We spent the year from 1992 to 1993 theorizing the methodology of the project and designing its logical structure. Then in 1993 we built the first small demonstration model of *The Rossetti Archive*” (12). Thus, McGann encountered many of the same problems with duplication in data structures and program components familiar to technologists in the process of modeling physical phenomena with code. “Because the entire system develops through the codex form, however, duplicate, near-duplicate, or differential archives appear in different places. The crucial problem here is simple: The logical structures of the critical edition function at the same level as the material being analyzed” (56). As a result, scholarly editing theory evolved with the new media: “When a book is translated into electronic form, the book's (heretofore distributed) semantic and visual forms can be made simultaneously present to each other" (56-57).

McGann concludes that the possibilities of hyperediting themselves create new problems while addressing existing problems, such as the markup tags for the documents to be contained in the archive, and the digital tools for implementing the markup. "Those of us who were involved with *The Rossetti Archive* from the beginning spent virtually the entire first year working at this problem. In the end we arrived a a double approach: first, to design a structure of SGML markup tags for the physical features of all the types of documents contained in *The Rossetti Archive* (textual as well as pictorial); and second, to develop an image tool that permits one to attach anchors to specific features of digitized images" (69). Buzetti and McGann see markup as a highly reflexive act, with oscillating indeterminacy like self-organizing systems; "as soon as a (marked) text is (re)marked, the metamarkings open themselves to indeterminacy" (67). As a consequence, McGann relates the pains taken in *Radiant Textuality* to annotate the many revisions to the encoding schemes he and his collaborators went through while engineering production of the *Rossetti Archive*, which for a long time was held as a model digital humanities project and electronic archive edition.

Moreover, the interesting suggestion by Buzzetti and McGann for researching autopoietic functions of social textualities via user logs dovetails nicely with software studies and projects for future digital humanities scholars. "This set of materials—the use records, or hits, automatically stored by the computer—has received little attention by scholars who develop digital tools in the humanities. Formalizing its dynamic structure in digital terms will allow us to produce an even more complex simulation of social textualities" (71). We may discern in these acts of rigorous scholarship practices readily apparent in software development communities, whose texts of source code are both the subject of study and object of continual revision.

There is another connection to make between textuality studies and programming studies. McGann characterizes predigital scholarly books as postmodern incunables. “At once very beautiful and very ugly, fascinating and tedious, these books drive the resources of the codex to its limits and beyond. Think of the Cornell Wordsworth volumes, a splendid example of a postmodern incunable. Grotesque systems of notation are developed in order to facilitate negotiation through labyrinthine textual scenes. To say that such editions are difficult to use is to speak in vast understatement. But their intellectual intensity is so apparent and so great that they bring new levels of attention to their scholarly objects” (79). Similarly, the grotesque, labyrinthine notations of early computer programming languages that have been long discarded along with the antique hardware on which they were developed beg for poetic theorization in electronic editions, with comparison to high-level languages in use today. Donald Knuth and Luis Pardo surveyed the evolution of high level programming languages for 1945 to 1957, commissioned for 1977 publication in the *Encyclopedia of Computer Science and Technology*.

Based on unpublished source materials, with explicit emphasis on how languages were developed, what human elements are expressed in each language, and appreciation of the amount of progress now built into the environment and regarded as self evident, their study seeks to foreground these details mostly ignored by computer scientists (“The Early Development of Programming Languages” 2). This history of the first decade of high level programming languages, now considered dead and primitive like Greek and Latin to humanists, yet due to the additive nature of built environment, is concretized within it. As they put it, “our story will take us up to 1957, when the practical importance of algebraic compilers was first being demonstrated, and when computers were just beginning to be available in large numbers. . . . The best languages we shall encounter are, of course, very primitive by today's standards, but they were good enough to touch off an explosive growth in language development" (2).[46]

To transition from textuality studies to media studies, I want to note that McGann promotes a range of digital practices from the conservative to bizarre when it comes to answering the question of how to do humanities with computers. "Whereas we thrive in a world of analogues and fuzzy logic, computers exploit a different type of precision. What if the point were not to try to bridge that gap but to feed off and develop it? Meditating that question is the recurrent object of this book's last five chapters. All move in pursuit of a new ground on which to build computerized tools that generate and enhance critical reflection. . . . Electronic or not, our tools are prostheses for acting at a distance" (103). It is obvious, then, that scholars create digital tools to refine and extend the practices they have developed from centuries working with print. He also recommends and demonstrates using computers to challenge the inherent biases these habits convey. "We will argue that concept-based interpretation, reading along thematic lines, is itself best understood as a particular type of performative and rhetorical operation" (106). For instances, Dante's *Convivio* provides a model for hermeneutics: "reading backward is a deformative as well as a performative program. . . . Coming before the historical period when prose gained its scientistic function, the *Convivio* is especially important: for it is also the work that models and licenses many of our most basic hermeneutic procedures" (109-110).

46 This piece was originally planned for coverage in the "Pioneers of Babelization" section of a separate chapter about these philosophical programmers that has been removed from the scope of this work and reserved for future study.

McGann dares to enter the forbidden zone of deformative scholarship also practiced by Barthes, Derrida, Ulmer, and O'Gorman. Indeed, Barthes contends that deformation is essential to myth: “the concept, literally, deforms, but does not abolish the meaning; a word can perfectly render this contradiction: it alienates it” (“Myth Today” 92). Hayles praises McGann's so-called 'experiments in failure' as examples of software that keeps revising itself, versus static literary texts. Deformation taken as a reading practice, emphasizing the importance of doing and making, is consonant with critical inquiry that is productive, “suggesting that practical experience in electronic textuality is a crucial prerequisite for theorizing about it” (*My Mother Was a Computer* 99). As McGann puts it, “deformative scholarship is all but forbidden, the thought if it either irresponsible or damaging to critical seriousness. . . . The reluctance shows, more interestingly, that interpreters—even radical ones—do not commonly locate hermeneutic vitality in the documentary features of literary works. Because meaning is assumed to develop as a linguistic event, critical deformance plays itself out in the field of the signifieds” (114-115). He invokes Della Volpe's notion of dialectical criticism, which is different from that of Hegel and Heidegger who reveal unknown knowns, to apply instead to *imaginations*. “Interpretive moments stand in nonuniform relations with each other so that the interpretation unfolds in fractal patterns of continuities and discontinuities. Besides realizing, perhaps, what we didn't know we knew, we are also led into imaginations of what we hadn't known at all. . . . Meaning is important not as explanations but as residue. It is what is left behind after the experiment has been run. We develop it not to explain the poem but to judge the effectiveness of the experiment we undertook” (129).

Similarly, Gregory Ulmer seeks to treat reading as if it were computing to allow open system, interpretant forms of cognition. He explains that “the question for AG [Applied Grammatology] concerns how the student might operate in accord with the hypomnemics of the electronic paradigm. At this stage of transition, the fields of knowledge, as represented in encyclopedias, textbooks, and the like, may be manipulated by the learner as if reading were computing. . . . AG distinguishes itself from the psychologisms of current reader-response subjectivism by concerning itself not only with the field of oriented possibilities (that which actually or phenomenologically occurs in the inner speech of a student) but with constructing connections among the systems in relation to the field of all possibilities” (*Applied Grammatology* 311). Looking ahead, deformations of software systems, described by Marino and others, such as stepping through processes and otherwise altering their normal temporal behavior, can likewise result in what Ulmer registers as “the dramatic exposure of subjectivity as a live and highly informative option of interpretive commentary” (116).

McGann ultimately envisions a human computer symbiosis in which humans do analog and computers digital thinking, the latter ignorantly performing deformations and submitting results for human consideration. “We want to see what textual possibilities have been forbidden or made nugatory by the original act of textual encoding—that is, by the decisive and particular text that stands before us. The random access procedures of digital technology can bring those possibilities to view. The fact that many will appear to us, at that point, as *impossible* nonsense is exactly what holds out such promise, on two counts. First, not everything tossed up by the computer will seem nonsensical, and besides, people will differ. Second, however we judge the results, they will inevitably clarify our own thoughts to ourselves by providing a set of contrasts to throw our thinking into sharper relief" (190-191). In his analysis, basic forms of digital life correspond to advanced self-conceptions of book culture.[47] This conclusion seems to foreclose on notions of emergence and co-constituted subjectivity that Hayles suggests, and I have already laid out as part of the post-postmodern network dividual cyborg. Therefore, I will to pass the baton from McGann and textuality studies to theoretical frameworks emerging from media studies, which more rigorously take into account the technological and social constraints and affordances of their subject matter, before focusing on the nascent disciplines of software studies and code studies.

Media Studies

There is a plethora of theorists under the banner of media studies, so I have selected a handful who seem aligned with contemplating the philosophy of computing and programming in the context of better knowing ourselves as posthuman cyborgs. McLuhan famously declared that the medium is the message, that the content of media is previous media, often turned into an art form. "Each new technology creates an environment that is itself regarded as corrupt and degrading. Yet the new one turns its predecessor into an art form. When writing was new, Plato transformed the old oral dialogue into an art form" (ix). Moreover, the effects of technological media "do not occur at the level of opinions or concepts, but alter sense ratios or patterns of perception steadily and without any resistance" (33). While many point out the effects of rapid

47 This is a description of real virtual production as intended by Castells, situating Mallarme, Rossetti, Sinburne, Morris, Dickinson and Whitman as similar artisans crafting virtual reality machinery, what Murray refers to as the holodeck, in the media available to them, or else their work (texts) manifesting the asymptotic limit of those fantasies symptomatically.

technologization on cultures that have not spent centuries under the sway of print, he notes “that is only the East-side story, for the electric implosion now brings oral and tribal ear-culture to the literate West. Not only does the visual, specialist, and fragmented Westerner have now to live in closest daily association with all the ancient oral cultures of the earth, but his own electric technology now begins to translate the visual or eye man back into the tribal and oral pattern with its seamless web of kinship and interdependence” (58). McLuhan argues that games are mass media, long before videogames, for they “are extensions, not of our private but of our social selves, and that they are media of communication. . . . Games are situations contrived to permit simultaneous participation of many people in some significant pattern of their own corporate lives” (216). Jumping forward a few decades, McGann argues that hypermedia are profane resurrections of once-sacred models of communication. “In the rediscovered Grotesque art of the Middle Ages was heard the metaphor—it deliberately mixed the first premonition of the famous proverb that would define the coming of the digital age a century later: the medium is the message” (xii). Interpretations veer toward the effect of media upon human perception and consciousness.

Fitting as they may be for theorizing Bauerlein's dumbest generation retreating into social media cocoons, these attempts to draw analogies between technological media and prior forms, as phenomenologists we cannot avoid the fact that, quoting Manovich, “today we are in the middle of a new media revolution—the shift of all culture to computer-mediated forms of production, distribution, and communication. . . . In contrast [to photography], the computer media revolution affects all stages of communication, including acquisition, manipulation storage, and distribution; it also affects all types of media texts, still images, moving images, sound, and spatial constructions" (*Language of New Media* 19). To better contextualize the innovation of the technological apparatus translating all existing media to numerical data, Manovich stakes out a historical media studies perspective he calls digital materialism, leaning heavily on cinema. "The advantage of placing new media within a larger historical perspective is that we begin to see the long trajectories that lead to new media in its present state, and we can extrapolate these trajectories into the future" (10).

Whereas text was already amendable to digitization, under numerical representation other media become programmable by digitizing continuous data through sampling and quantization. This releases postmodern analytics from their awkward fit with static media, and makes it easier to speak of streaming audiovisual phenomena as texts. The language of cinema is moving images rather than printed text, and it serves better to articulate key trends shaping new media: modularity, automation, variability, and transcoding. Manovich takes pains to identify the semiotic bias for discrete representation as morphemes from trends arising in the Industrial Revolution. "Not surprisingly, modern media follows the logic of the factory, not only in terms of division of labor as witnessed in Hollywood film studios, animation studios, and television production, but also on the level of material organization" (29-30). Manovich extends Benjamin's preliminary statements about the work of art in the age of mechanical reproduction by retelling the history of cinema to make the point again and again that new media developed out of not only mechanical production techniques but a cultural milieu becoming accustomed to commodities, utilities, commercialism, marketing, and so on.

Many theorists including Barthes, Derrida, Ulmer, Landow, and Hayles appeal to the bite-size literary units typical of postmodern writing styles as reflecting the impact of industrialization on language. “Like Barthes, Derrida conceives of text as constituted by discrete reading units,” Landow states, developing his notion of the *mourceau* in connection to hypertext. Continuing, “this *mourceau*, adds Derrida, is always detached, as its name indicates and so you do not forget it, with the teeth, and these teeth, Ulmer explains, refer to quotation marks, brackets, parentheses: when language is cited (put between quotation marks), the effect is that of releasing the grasp or hold of a controlling context. . . . Derrida, the experience of hypertext shows, gropes toward a new kind of text: he describes it, he praises it, but he can only present it in terms of these devices here—those of punctuation associated with a particular kind of writing” (*Hypertext 3.0* 54).[48] Derrida experimented with new textual forms suggested by industrial, proto-digital technologies and their implications, though remained a writer of books and papers. “Derrida, more than any other major theorist, understands that electronic computing and other changes in media have eroded the power of the linear model and the book as related culturally dominant paradigms. . . . The problem, too, Derrida recognizes, is that one cannot tamper with the form of the book without disturbing everything else in Western thought” (67).

48 Bolter continues Landow's meditation on the fit between deconstruction and hypertext, noting that deconstruction, although playful, require seriousness, which complicates criticism of hypertext works. “The hypertext authors since 1980s have in general created playful, allusive hypertexts that do not take themselves too seriously, as a printed text seems inevitably to do. Why would anyone want to deconstruct a work entitled *Uncle Buddy's Phantom Funhouse?* Such hypertexts are not hostile to criticism, but are instead self-referential and incorporate their own critique. . . . Electronic writing seems then to accept as strengths the very qualities - the play of signs, intertextuality, the lack of closure - that the poststructuralists posed as the ultimate limitations of literature and language" (*Writing Space* 182-183).

While we may consider *Archive Fever* to engage in media studies, I am arguing that Derrida must be set aside. As Hayles reminds us, print is now an output form of digital data rather than a separate medium. We have indeed tampered with the book form. “This engagement is enacted in multiple senses: technologically in the production of textual surfaces, phenomenologically in new kinds of reading experiences possible in digital environments, conceptually in the strategies employed by print and electronic literature as they interact with each other's affordances and traditions, and thematically in the represented worlds that experimental literature in print and digital media perform” (*Electronic Literature* 159).[49] Hayles' self-proclaimed styles of media studies over the years include Comparative Media Studies and Media Specific Analysis, both of which seek to discern the affordances and constraints of different textual surfaces, phenomenological discovery of new reading styles, and so on (“Print is Flat, Code is Deep”; *Writing Machines*).

As an update to Derrida for the digital era, Bogost presents *unit analysis* “for the general practice of criticism through the discovery and exposition of unit operations at work in one or many source texts. . . . Each medium carries particular expressive potential, but unit analysis can help the critic uncover the discrete meaning-making in texts of all kinds” (*Unit Operations* 15). Multiple small pieces relate to Derrida's morsels, and Bogost distinguishes this breezy, aphoristic perspective from what he calls systems operations. “In literary theory, unit operations interpret networks of discrete readings; system operations interpret singular literary authority. In software technology, object technology exploits unit operations; structured programming exhibits system

49 Contrast the image of Derrida musing over his portable Macintosh in *Archive Fever* with Kittler, who dove into the technical details of electronic and optical media.

operations. . . . In effect, the biological sciences offer an especially salient window into the development of unit operations. . . . In general, unit operations privilege function over context, instances over longevity” (3-4). It's easy to adopt the unit analysis style and eschew surveying an idealist, system level that formerly unified everything in a single explanatory and metaphysical rubric. Quoting Landow again, “this hypertextual dissolution of centrality, which makes the medium such a potentially democratic one, also makes it a model of society of conversations in which no one conversation, no one discipline or ideology, dominates or founds the others. It is thus the instantiation of what Richard Rorty terms edifying philosophy, the point of which is to keep the conversation going rather than to find objective truth” (123). The alignment of poststructuralist and postmodern discourses with the modularized, decentralized aspects of new media yield a putatively comfortable place for keeping a whole lot of conversations going.

An alternative approach to media studies largely ignores the specific content and focuses instead on the apparatus with technical rigor and critical attention to the social construction of technology. Kittler is well known for taking this position. Writing at the end of the twentieth century, he declares “under the conditions of high technology, literature has nothing more to say. It ends in cryptograms that defy interpretation and only permit interception. Of *all* long-distance connections on this planet today, from phone services to microwave radio, 0.1 percent flow through the transmission, storage, and decoding machines of the National Security Agency (NSA), the organization succeeding SIS and Bletchley Park. By its own account, the NSA has accelerated the advent of the computer age, and hence the end of history, like nothing else” (*Gramophone Film Typewriter* 263). Before arriving at this end of history in which media converge and forever circulate in electronic circuits, sense perceptions themselves had to be fabricated that were amenable to machine processing. Kittler traces the development of technologically enhanced perception through his study of gramophone, film and typewriter. “But these sense perceptions had to be fabricated first. For media to link up and achieve dominance, we need a coincidence in the Lacanian sense: that something ceases not to write itself. Prior to the electrification of media, and well before their electronic end, there were modest, merely mechanical apparatuses” (3).

Kittler's method is more subtle than presenting historical narratives that many find repugnantly deterministic. His phenomenological method resembles psychoanalysis in the sense that he discerns unconscious reflections about the nature of particular media from the style and content of particular works: hysterical traces through which the nature of technological media reverberates, what Thrift calls the technological unconscious. Thus he reasons that “we can provide the technological and historical data upon which fictional media texts, too, are based. Only then will the old and the new, books and their technological successors, arrive as the information they are. Understanding media despite McLuhan's title remains an impossibility precisely because the dominant information technologies of the day control all understanding and its illusion” (xl). Phenomenology seems doomed to operate within the confines of the mechanisms it seeks to discern. The truism that our understanding of media is affected by the media through which perception and communication passes forces indirect methods.

Kittler diverges from culture studies based approaches to media by invoking the requirement of understanding their science and design, urging theorists to look beyond commodity media to military applications as well. “The only thing that remains is to take the concept of media from there in a step also beyond McLuhan to where it is most at home: the field of physics in general and telecommunications in particular. . . . Second, the consequence of employing the media concept of telecommunications is that media studies cannot be limited solely to the study of media that (to be brief and clear) have a public, civilian, peaceful, democratic, and paying audience" (*Optical Media* 32). Yet his analysis seems too steeped in the trope of Heraclitus that war is the father of all things. By declaring that high fidelity stereo sound reproduction evolved as a misuse of military equipment, he invites and receives criticism from many fronts for technological determinism, the very sort the SCOT rallies against.

In this dismissal it is all to easy to reject the important contribution Kittler makes to media theory: his insistence that the so-called philosophers of computing be as well versed in electronics as they are in classical Greek. Knowing the difference between electricity and electronics, for example, is a precondition to understanding the science behind high speed switching and control at the core of stored program machinery. So he expects his readers to agree with him that "Braun's tube was not crucial for film and radio technology, however, but rather another tube variant: the so-called triode. . . . Two inputs were needed along with a general ground return, and it was therefore called a triode or three-way in the artificial Greek of technology. . . . Thus, the electron tube first decoupled the concept of power from that of physical effort. . . . negative feedback can be generated by leading the output signal, which for physical reasons is always delayed for fractions of a microsecond, back to the control circuit" (192).

Toward the conclusion of *Gramophone, Film, Typewriter* diagrams of a Z80 microprocessor circuit and standard CPU accompany his brief description of stored program electronic computer, which now captures every possible medium, fulfilling the deterministic

Laplacian universe in finite-state machines. As read by Hayles, it also promises illuminations beyond human manipulation (fortuitous deformations) that inaugurate post postmodern subjectivity. “That's all. But with sufficient integration and repetition, the modular system [of the microprocessor] is capable of processing, that is, converting into any possible medium, each individual time particle of the data received from any environment. . . . Every microprocessor implements through software what was once the dream of the cabala; namely, that through their encipherment and the manipulation of numbers, letters could yield results or illuminations that no reader could have found” (244-247).

The contours and significance of Kittler's media theory are better described for English readers by others than Kittler himself. According to Sybille Krämer, media are defined as culture techniques allowing selection, storage and production of data and signals (93). The default media epochs are alphabet, press and computer. “Analog media and optical-technological media in particular mark the beginning of a development that ends with digitization and the computer. In the age of handwriting and the printing press, all forms of writing are bound up in a symbolic universe which in its most basic variant is that of everyday speech to select, store, and produce the physical realities themselves” (94). Significantly, technological media “allow one to select, store, and produce precisely the things that could not squeeze through the bottleneck of syntactical regimentation in that they are unique, contingent, and chaotic” (94).

Freed from the constraints of the symbolic and strictly brain-bound perceptual phases, Kittler's media analysis can be orthogonal to aesthetics and sensibility. “His concept of media continually attempts to speak about the realm of literary studies in a way that avoids using distinctions such as 'understanding', 'interpretation', 'meaning', 'referent', or 'representation', terms that are integral to the vocabulary of literary studies. . . . Kittler is thus concerned not with a media analysis that is diametrically opposed to meaning, but rather with a practice of writing about media in which concepts such as sense and sensibility are no longer relevant” (94). In particular, the ability of technological media to store and process 'real time' as events, where in alphabetic and print media time is implicit but always manufactured in the act of perception, means that “*time itself becomes one of several variables that can be manipulated*. . . . *Data processing becomes the process by which temporal order becomes moveable and reversible in the very experience of space*" (96).

As Krämer interprets Kittler, discourse analysis belongs to the historical era of alphabet and press. “As soon as other types of discourse networks emerge with technological, analog media, then an archeology of present forms of knowledge can no longer be practiced by discourse analysis but must rather be taken over by technological media analysis” (97). This change in the operation of media to a performative, autonomous, autopoietic substrate makes it more obvious that media are production sites of data overdetermining what may come to presence, a presence with traces of symptoms that can be detected in discourse systems, as Kittler masterfully demonstrates in the examples given in *Gramophone, Film, Typewriter* and his essay “Dracula's Legacy.”

Time axis manipulation is Kittler's preferred term for describing how data processing techniques “use a spatial means to create possibilities of ordering the things *differently* that are etched into this spatial ordering. . . . Wherever something is stored, a temporal process must be materialized as a spatial structure” (99). The gramophone is a simple, predigital example. Transductions by operations of technological media afford new media effects like reversing temporally sequenced events, which, as others point out, affects pitch and other phenomena impossible to convey by manipulating text. Such are possibilities when the real is saved in the age of technical reproduction even before computers. With electrical and electronic storage media, the Fourier method does for the real of signals what the Greek alphabet did for symbolic language, permitting anything that can be sampled to be stored, reproduced, and transformed. Decoding the real with machines releases meaning from discursive subject to which Hayles feels postmodernism too rigidly adheres.

Technological media entail what David Berry and Ian Bogost call *double mediation*, which means that not only does one media contain another, as McLuhan pointed out, but the funnel of the signifier is supplanted by the real machinery of encoding and decoding that produce it. To Krämer this is the what Kittler means by media convergence. “The binary system provides a universal key that allows one not only to translate each of the numerous formats of image, sound, and textual media reciprocally, but also, and at the same time, to traverse the symbolic-technological boundaries of the epoch of alphabetic writing. . . . The computer connects all of these media, in that it incorporates their input and output into a mathematical procedure of digitalized signal processing with microsecond rhythms” (102-103). This ontology of switchable existences is Kittler's reductive technological ontology: only that which can be switched, i.e., encoded and decoded via some mechanism of time axis manipulation, exists at all. Moreover, technological media operate in higher frequency ranges where hearing and sight disappear. “Every type of phenomenology loses its foundation. Kittler's critique of hermeneutic sense-orientation encompasses phenomenological strategies. Both of these are not only falsified but they also become historically obsolete with the development of technological media” (106).

Time becomes the universal form of technological accessibility, yet one in which humans are not the key players.

Krämer concludes that Kittler's take on media studies ought to be considered a form of digitalized existentialism. “The culminating points within the tradition of literary studies . . . are transformed into the absurd with the binary code that cannot be expressed by humans, and with modern data processing as a textile that cannot be read by human eyes. This 'absurdity' can be understood in the sense of Kierkegaard as a paradox of the historical, which stands in stark opposition to human logic and sensibility, or in the sense of Camus, who counter-intuitively lets the world remain mute to human questions” (108). Nonetheless, Kittler demonstrates his allegiance to the tradition of literary studies by making countless references and connections between classical literature, music, and art to his engineering observations about computer architecture and optical media. Therefore, his role in my theoretical framework is not to call for the eviction of critical theories and textuality studies grounded in the literary and symbolic registers of natural mother tongues, but rather to insist as a precondition for any putative philosopher of computing to incorporate a sophisticated practical knowledge of electronic technology alongside the standard humanities equipment so that the absurdity of his digitalized existentialism is not all encompassing and interminable.

Social Construction of Technology

The next component of my theoretical framework adds empirical rigor to approaches that are largely built upon textual connections and philosophical ideas. Electric media as an extension of the central nervous system alters the human subjectivity built from long acculturation to oral and literate cultural practices. Kittler in particular was a literary theorist before he was a media scholar, and the influence of Heidegger, Lacan and others who pepper their philosophical insights with cherry-picked technical references are evident. So although he tries to speak with the authority of an engineer's insider knowledge, he often sounds like someone dabbling with the terms of various technological disciplines to make his larger philosophical points. Such is the impression from viewing video lectures at the European Graduate School in the years before his death in 2011.[50]

The involvement of large organizations deliberately planning, designing, manufacturing, regulating, marketing, and training the masses of humanity in the use of new media differentiates them from prior forms that emerged less formally: at least this is the typical conception of the technological media. It is advisable, therefore, to turn next to a closer examination of the *social* construction technology, so as to eventually critically investigate the constellations of particular computing technologies that constitute the current milieu in which the human situation seems to be regressing.

A suppressed section on the philosophy of technology will be briefly summarized here to segue into its present incarnation as Social Construction of Technology (SCOT), or Science and Technology studies (STS) as Deborah Johnson calls it. Don Ihde notes that the philosophy of technology itself is a latecomer to American academia. “Interestingly, in Europe this included at

50 He will, for instance, state that we must understand the difference between a switch and a transistor, and that he will differentiate electricity from electronics, but never really gets into the details in his lecture (“The March of Technology”). Kittler is following the tradition of Hegel, Lacan, Foucault, and others who appear to be presenting a large quantity of disparate observations to arrive at a position, where they instead have arrived at a philosophical position and are presenting evidence after the fact.

least one foray into *Tecknikphilosophie,* a book by the neo-Hegelian, Ernst Kapp, in 1877! Yet, it was to be nearly 100 years later that philosophy of technology became a recognizable characterization within philosophy in North America" (14). Traditional philosophy favors Descartes over Bacon, for which Ihde argues in consequence the instrumental and technical sides of scientific practice get overlooked. They are overlooked in favor of formal logic, which is ignorant of material modes of production shaped by technologies, as well as the roles of cultural and social knowledge in forming (informing) technologies (29). In his broad historical sweep, Ihde contends that the Anglo-American analytic philosophy of science tradition is unconcerned with technology, whereas phenomenological, pragmatist, and political traditions foreground it, namely praxis philosophers like Marx, leading to Kuhn and modern history and philosophy of science and technology studies (46).

While a paradigm shift may still be in progress for philosophy proper, numerous companion disciplines have arisen that foreground these concerns. When Ihde describes the designer fallacy, where "only sometimes are technologies actually used (only) for the purposes and the specified ways for which they were designed," (116) and the management of high technology settings as "much more like controlling a political system or a culture than controlling a simple instrument or tool" (117), he is exercising common themes in SCOT scholarship. My point is that since the philosophy of computing is itself a barely formed discipline, it may be able to avoid some of the growing pains suffered by Anglo-American philosophy's long adolescence in the cocoon of formal logic by forming itself in multiperspectival, heterogeneous theoretical lines of flight from the outset.

As Bijker, Hughes and Pinch comment in the introduction to *Social Construction of Technological Systems*, “a characteristic that all these approaches share is the emphasis on thick description, that is, looking into what has been seen as the black box of technology (and, for that matter, the black box of society)” (5). Edwards affirms that “as Pinch and Bijker themselves have noted, few studies have managed fully to engage the relationship between the meanings of scientific facts or technological artifacts and their sociopolitical milieu” (34). To Feenberg, social meaning and cultural horizon are hermeneutic dimensions of technical objects, despite their inauthenticity in the eyes of philosophers like Heidegger (“Democratic Rationalization” 656). “Devices, Heidegger complains, race toward our goals and lack the integrity of his favorite jug or chalice. But by what rights does he make this summary judgment on the very things that surround him? Devices are things too. Modern and technological though they may be, they too focus gathering practices that bring people together with each and with earth and sky, joining them in a world” (*Questioning Technology* 196). This expanded hermeneutic dimension has been illustrated by Richard Johnson's diagramatic models representing “a circuit of the production, circulation and consumption of cultural products. . . . if we are placed at one point in the circuit, we do not necessarily see what is happening at others. . . . Processes disappear in results. . . . the conditions of their production cannot be inferred by scrutinizing them as texts. . . . we cannot predict these uses from our own analysis” (46-47).

In Edwards' view, techniques, technologies, practices, fictions, and languages have formed the closed-world discourse where a dominant narrative has been promulgated to provide a singular, deterministic historical trajectory to technological artifacts, especially computing machinery (15). As consequence of this preference for closed world discourses, as I read Latour, the critical stance has reached a crisis because naturalization, socialization and deconstruction each have attracted adherents producing powerful theories – E.O Wilson, Pierre Bourdieu, and Jacques Derrida are emblematic figures – but none of these positions can be combined with the others. “Is it our fault,” Latour asks in *We Have Never Been Modern*, “if the networks are *simultaneously real, like nature, narrated, like discourse, and collective, like society*?” (6). Therefore, it takes a special sort of researcher, which he describes as “hybrids ourselves, installed lopsidedly with scientific institutions, half engineers and half philosophers, '*tiers instruits*' [educated arbitrator] without having sought the role, we have chosen to follow the imbroglios wherever they take us. . . . Yet this research does not deal with nature or knowledge, with things-in-themselves, but with the way all these things are tied to our collectives and to subjects. We are talking not about instrumental thought but about the very substance of our societies” (3).

The anthropological and ethnological methods Latour prefers tackle everything at once. Thus he takes the argument between Boyle and Hobbes from its philosophical and political points to focus on the air pump the former used to perform experiments, mirroring the approach Hayles takes in her study of cybernetics that homes in on the Macy Conferences and its air pump, the neuron model. “For the first time in science studies, all ideas pertaining to God, the King, Matter, Miracles and Morality are translated, transcribed, and forced to pass through the practice of making an instrument work” (20). Interestingly, Latour argues that the air pump itself, and instruments in general, never reach the universality of physical laws. Rather, “its network is extended and stabilized. . . . By following the reproduction of each prototype air pump throughout Europe, and the progressive transformation of a piece of costly, not very reliable and quite cumbersome equipment, into a cheap black box that gradually becomes standard equipment in every laboratory, the authors bring the universal application of a law of physics back within a network of standardized practices” (24).

We could likewise trace the development of the personal computer from cumbersome oddities to the cheap black boxes they are today. The point is that scientific instruments and computers exemplify what Latour calls *quasi-objects*, “much more social, much more fabricated, much more collective than the 'hard' parts of nature, but they are in no way the arbitrary receptacles of a full-fledged society. On the other hand they are much more real, nonhuman and objective than those shapeless screens on which society for unknown reasons needed to be 'projected'” (55). Even computational objects are not pure simulacra. Latour's hyperbolic rhetoric serves to maintain this hybrid, syncretic position of half-engineer, half-philosopher maintaining “the notion that things themselves have a history” (70). Subsequent sections will show how the nascent disciplines of Software studies and Code Studies apply the same consideration of social and historical determinations to machine texts and other assemblages, about which McGann contends “the truth is that all such works are special because they call attention to a crucial general feature of textuality as such: its social and historical determinations” (166).

SCOT acknowledges the seamless web of society and technology. It combines three approaches of social constructivism, whose key concepts include interpretative flexibility, closure, and relevant social groups, with Hughes' systems metaphor, and actor networks. Bijker, Hughes and Pinch insist they “are concerned here with only the recent emergence of the sociology of scientific knowledge. Studies in this area take the actual content of scientific ideas, theories, and experiments as the subject of analysis. This contrasts with earlier work in the sociology of science, which was concerned with science as an institution and the study of scientists' norms, career patterns, and reward structures” (18).[51] Focusing on the content requires examining the many iterations of a particular invention, which often gains its current status retroactively. “Historians of technology often seem content to rely on the manifest success of the artifact as evidence that there is no further explanatory work to be done [for example, Bakelite plastic]. . . . However, a more detailed study of the development of plastic and varnish chemistry, following the publication of the Bakelite process in 1909, shows that Bakelite was at first hardly recognized as the marvelous synthetic resin that it later proved to be” (22-24).[52]

Bijker's famous bicycle study reveals that quasi-linear development is a retrospective distortion, recommending instead a multidirectional model by studying development process as the alternation of variation and selection. Furthermore, interpretative flexibility must be shown in design as well as reception and use. In order for an interpretation to solidify as the accepted narrative that retrospectively historicizes an object, rhetorical closure must occur within various social groups. “The key point is whether the relevant social groups see the problem as being solved. In technology, advertising can play an important role in shaping the meaning that a social group gives to an artifact” (44).[53] For the study of computing machinery, networks, and software, the SCOT approach opens a body of literature dominated by monolithic, heroic narratives such as Levy's *Hackers* and Freiberger and Swaine's *Fire in the Valley* to mulitidirectional, often

51 I observed a parallel shift in the subject of analysis of programmers and managers from norms and career patterns in Kraft to everyday practice in the ethnologies presented by Rosenberg and Mackenzie.

52 Compare to Manovich in *Software Takes Command* on why there are no studies of cultural software, implying asymmetry between state of the art and prior versions in addition to commercial failures.

53 We could give the example of advertised computer security solutions like virus scanners and firewalls to insecure operating environments as rhetorical closure.

contested accounts like Abbate's *Inventing the Internet*. This will be demonstrated in the next section touching on histories of computing, networking and software.

To achieve the goal of a multidirectional perspective, SCOT researchers strive to turn the study of technology into a sociological tool by examining hypotheses and arguments made by those Michel Callon calls engineer-sociologists. “To bring this reversal about, I show that engineers who elaborate a new technology as well as all those who participate at one time or another in its design, development, and diffusion constantly construct hypotheses and forms of argument that pull these participants into the field of sociological analysis” (83). He is, in part, challenging the ability to distinguish the distinctly technical from any economic, cultural and commercial logics affecting technological change that other theorists take for granted.

Callon describes a case study of the VEL electric car initiative in France that highlighted contested designs and visions between engineers at EDF and Renault, as well as impact of non-human network actors like battery components. EDF depicted the social position of urban post-industrial consumers turning away from internal combustion engines and downgrading the status of private automobiles as consumer objects. First he invokes the perspectives of two sociologists: Touraine, who argues that in post industrial society key class conflict happens between technocrats and consumers, and Bourdieu, for whom consumption is the key facet of upper and lower class competition. Thus he initially casts the future of automobile in terms of Touraine versus Bourdieu, paralleling identifiable positions taken by VEL and Renault engineers. What differentiates engineer-sociologists from classical sociologists is the former make heterogeneous associations ranging over actor networks, whereas the latter remain too narrowly focused on contributions of human actors.

Actor networks in this case study include entities ranging from electrons and catalysts to industrial firms and consumers. Callon argues their dynamics can be explained by mechanisms of simplification and juxtaposition, which are implicit in the diachrony in synchrony model I propose at the end of this chapter. Simplification of an infinite reality to limited associations of discrete entities with well defined attributes is a basic engineering stance. Simplification masks unknown sets of entities drawn together by known entities in the network.[54] Juxtaposition of heterogeneous elements that guarantee proper functioning of objects transcend restricted analytic categories. “Whatever their nature, what counts is that they render a sequence of events predictable and stable. . . . Each element is part of a chain that guarantees the proper functioning of the object. It can be compared to a black box that contains a network of black boxes that depend on one another both for their proper functioning as individuals and for the proper functioning of the whole” (95). Extremely complex, cascading operations yield a durability of simplifications sustaining the actor network at each point, yet this latent instability provides conditions leading to transformations, which can be discerned by testing resistances. Callon argues that classical sociologists are unable to take such heterogeneous associations into account, thus both Touraine and Bourdieu are susceptible to his critique, and Bourdieu's interpretation was only right about VEL by chance.

The new methodological tool Callon proposes follows the innovators in a concrete analysis, for they often develop their own sociological theories that are evaluated by empirical outcomes like market share and profits before scholarly reflections are applied. Additionally, the

54 Callon notes that often the elements hidden by simplification are only revealed when they are brought into controversy through what Latour calls a trial of strength. Berry uses similar language to describe the conditions under which particular clusters of software source code are legitimized, what I call working code.

actor network as a style of sociological study gives the maneuvering and freedom engineers enjoy. “To transform academic sociology into a sociology capable of following technology throughout its elaboration means recognizing that its proper object of study is neither society itself nor so-called social relationships but the very actor networks that simultaneously give rise to society and to technology” (99). Software studies and Code Studies apply similar considerations of social and historical determinations to machine texts and other assemblages. As I build my theoretical framework and methodology, I seek out the insights of what I call philosophical programmers—the engineers, computer scientists, elite hackers, and everyday programmers who perform the role of “society in the making,” the title of Callon's text, alongside those who examine it from the spectator position.

An additional influence of SCOT on my theoretical framework is unrelenting attention to what Brown and Duguid call the social life of information. Their studies of office life reveal a combination of technological frailty and social resourcefulness, and stress the importance of living knowledge in organizations. “Most systems, amalgams of software and hardware from different vendors, rely on social amalgams of this sort keep everything running. . . . Infoenthusiasts, however, tend to think of these the other way around, missing the role of the social fabric and assuming that individuals in isolation can do it all” (77). Besides providing a different explanation of why telecommuting has not supplanted the traditional office setting than Castells gives, it strips information technology of some of its solipsistic bias that assumes a well-designed system is self-sustaining without a network of users supporting each others' efforts as well. This point reverberates through their critique of process reengineering. Local workplace cultures that promote banal chatting through lateral links are discouraged by process reengineers because they do not see the value of their linkages.[55] They find innovations are often hidden due to processes and forms; “by subordinating practice to process, an organization paradoxically encourages its employees to mislead it. Valuing and analyzing their improvisations, by contrast, can be highly informative” (110). They invoke Michael Polanyi's distinction between explicit and tacit dimensions to reinforce need for practices within community of practitioners to produce actionable knowledge in people, not just formal documentation.[56] Finally, Brown and Duguid decry the loss of collective memory from downsizing because organizational knowledge inheres more in people than databases (122).

While seemingly unrelated to the phenomenology of code, this perspective is repeatedly uncovered in ethnographic studies of software development communities and histories of large technological projects like the early Internet. What is the point? To extend code and technical protocols into their communities of practice, as scholarly textual analysis involves the critical apparatus whose notation often dwarfs the putative content, lest analysis focus on individual blocks, ignoring the surrounding apparatus and living communities sustaining them.

Histories of Computing, Software and Networking

Recalling a thread from the first chapter, although Bauerlein blames lax teaching and parenting for allowing the dumbest generation to remain absorbed in adolescent social media

55 Gee and Spinuzzi also emphasizes community of practice for situated learning.

56 Interestingly, the model for collective intelligence I term 'Foucault's dust' mistakenly identifies the documents and records as objects of explicit knowledge, ignoring both the operational practices of these knowledge users and the necessary involvement of their tacit understandings in managing the continuation of overall collective sensibility.

cocoons, he and many other critics point to consumer computer technologies as the key enabling factor. Especially because computing does so much more now than perform numeric calculations via effective methods, from the standpoint of the end results presenting multimedia, telecommunications, and commerce, which Manovich categorizes as cultural software, it is crucial to seek understanding of their inner workings lest their schematism of perceptibility remains veiled. David Berry, establishing the preconditions for any philosophy of software, argues “if code and software is to become an object of research for the humanities and social sciences, including philosophy, we will need to grasp both the ontic and ontological dimensions of computer code” (28). The only way code can be abstracted from the equipment that executes it is to ignore that material basis. Consequently, Berry follows other theorists by walking his reader through an abridged history of computing in order to situate the ontological and ontic dimensions of software at the heart of his work.

The timeline progresses from basic mechanical processes to the stored program computer, then multiprocessing and internetworking, to the present technological era called Web 2.0. Berry also reviews shifting notions of calculation and computation from that of mathematical engine to symbolic processing: “with theoretical contributions from Alan Turing (1936), Emil Post (1936), and Claude Shannon (1938) the notion of calculation moved from a problem of arithmetic to that of logic, and with it the notion that information can be treated like any other quantity and be subjected to the manipulations of a machine” (47-48). Likewise, the approach Castells takes in claiming “the brief, yet intense history of the information technology revolution has been told so many times in recent years as to render it unnecessary to provide the reader with another full account” permits him to refer to the work of historian Paul Ceruzzi among others for the well exercised history of the IT revolution, of which he gives a condensed version, concluding that 1990s networking has been decisive, then discussing the creation of the Internet (38).

Is this history likely to become part of general education, or will it be needed for introductory courses grounding digital humanities and philosophy of computing? A historical background and exposure to psychological and sociological studies hopefully expands folk wisdom and reveals biases about computing practices based on lifetime use situated within a cultural and technological milieu. A reason for examining histories besides developing a broad and deep familiarity with the origins of the very common is to discover points where ideas taken for granted today were novel, and as Hayles does in her study of cybernetics to supplement the default historical narrative with the sense of contingency upon which so many decisions and events hinged. As we enter working code places these advances can be revisited as one solution among many in vast fields of possible configurations of technogenesis and, consequently, synaptogenesis as well. I will likewise not attempt an exhaustive survey, but rather present the studies that have informed my theoretical framework, and reflect concerns of critical theory, media studies, and the social construction of technology in addition to paying heed to historical practices of rigorous scholarship and treatment of primary and secondary sources.[57]

57 The criticism could be raised that my selection of an English language corpus itself biases my background understanding in favor of Anglo-American narratives focusing on technologies that arose and prospered primarily in the United States, ignoring what was happening in other regions during the same time period. Campbell-Kelly notes this bias in both of his books, but does cover developments in other European countries; what happened in the former Soviet Union, the only other large electronic technology innovator, has yet to appear in mainstream English language publications.

In their famous 1946 report to the U.S. Army Ordnance Department "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," authors, primary investigators and designers of the Electronic Discrete Variable Automatic Computer (EDVAC) Burks, Goldstine and von Neumann admit that "it would take us much too far afield to discuss these questions at all generally or from first principles. We will therefore restrict ourselves to analyzing only the type of code which we now envisage for our machine" (3). By acknowledging that starting from first principles to conceive code would be a digression, electronic computing is philosophically shortchanged from the start.[58] Indeed, it is primarily philosophers and historians, not engineers, who ask that grounding question, *what is computing*?

Regarding the etymology and history of the use of the word 'computer', its Latin root *computare* means to reckon with, "to reckon, to think or to section, to compare the pieces" (Berry 11). Mario Aloisio states that "according to Borst, the word *computare*, which meant to reckon up, or to count on one's fingers, was already in use in early Roman times. This word frequently accompanied the word *numerare*, which had a similar meaning. Later, the word *calculare* was added to indicate counting of numbers with beads (or pebbles)" (42). Aloisio's interesting paper connects the origin and use of the word with the calculation of the Easter day, which was an arduous task based on the rules laid down by the Nicene council in 325. "The

58 I am not claiming that von Neumann was not deeply thoughtful about the philosophy of computing, as his voluminous notes and lectures on automata theory reveal. My point is that these works are seldom cited in general introductions to computing, especially in the humanities. As I claim in developing my methodology, there is a vast, unexplored territory of writings by philosophical programmers like von Neumann, Turing, Licklider, and so on whose work gave form to the architectures, programming languages, and design conventions buried in the microstructure of everyday computing machinery in use today.

general consensus among Christians was that Easter should be celebrated on a Sunday and, importantly, on the Sunday after the feast of the Jewish Passover. Passover is based on the lunar cycle; consequently, the date of Easter was inextricably linked with the moon. To calculate this date therefore required almost the impossible: an accurate determination in advance of the movements of the sun, earth, and moon" (42-43).

Of course, these ancient computers were human beings calculating in accordance with effective methods, not mechanical devices. Aloisio contends that the first reference to a mechanical computer is a fictional fantasy. "In part III of *Gulliver's Travels*, Swift refers to another 'computer' with the aid of which anyone would be able to master all the arts and sciences. This must be one of the earliest instances when the word was used—by the same author and in a short space of time—to refer both to a machine and a person" (45). Logarithms and calculating machines likely developed for doing trigonometry for navigation, especially determining longitude, and compound interest for accounting. However, mechanical devices at best aided large numbers of human computers. Divide and conquer and division of labor instead became key characteristics of humans doing computing activities. "As early as the 1920s, the term computing machine had been used for any machine that did the work of a human computer, that is, that calculated in accordance with effective methods. With the onset of the first digital computers, this term gradually gave way to computer" (47).

Other philosophical approaches to computing bypass these etymological and hermeneutic studies in favor of analytic approaches, often demonstrating a seamless transition from philosophical logic to computer architectures and machine languages. Their shortcoming, I argue, is that these models do not scale for interpreting network phenomena. Copeland's article

“What is Computation?” develops a labeling scheme that formalizes architecture specifications to constitute code. While able to develop an axiomatic specification of machine architecture for a simple, von Neumann machine (which will be discussed below in its historical context), it is questionable whether such approaches are still useful in massively distributed networks full of shoddy but adequate code. Luciano Floridi uses a similar approach in *Philosophy and Computing* of deriving the basic architecture of von Neumann Machines (VNMs) via mathematical logic, and spending many pages working through mathematical proofs. Yet in spite of declaring his approach to be 'critical constructivism', he answers the question of why software programs written for Mac and PCs, both VNMs, are incompatible using the logical reasoning that arrives at the simplest schematics of now ancient computers. “The same architecture can have different hardware implementations. . . . On the other hand, since each type of CPU recognizes a certain set of instructions, if two microprocessors implement different ISA [information structure architecture] then their software will not be compatible. This is why we cannot use software written for a Macintosh with an IBM-compatible and vice versa, although both are VNMs. It is like saying that a whale and a dog are both mammals but cannot eat the same food” (55).

Floridi does not consider that incompatibilities go beyond information structure architectures, having social and legal components that may be concretized in their design. Nonetheless, he argues forcefully that computer technology is all about ontics, instantiating poststructuralist and postmodern theoretical ideas of interest to the humanities. “First, we can now abandon the common view that hypertext (the conceptual structure, not the actual products) is simply an epistemological concept. . . . As the system of relations connecting the DIKs [data-information-knowledge], hypertext is the very backbone of the infosphere and significantly contributes to its meaningfulness in an ontical sense, i.e. by helping to constitute it. . . . Second, the space of reason and meaning including the narrative and symbolic space of human memory is now externalized in the hypertextual infosphere, and this brings about four more consequences concerning the rhetoric of spatiality” (129).

Aloisio's conclusion that “considering what modern computers are now capable of doing, the word that describes them has, paradoxically, almost become a misnomer” (48) is shared by Manovich, who prefers to talk about the operations of cultural software. However, nearly all of the consumer computing devices in use today share the same basic structural components that were laid out by Burks, Goldstine, and von Neumann in their report. This first page on the principle components of the machine should be required viewing by any student of computing, for it articulates the essentials of traditional, mainstream computer organization present in everyday devices: memory, control, arithmetic, input, and output. “Inasmuch as the completed device will be a general-purpose computing machine it should contain certain main organs relating to arithmetic, memory-storage, control and connection with the human operator” (1).

Like Turing's universal tape-based model, they have the ability to instruct an all-purpose machine to carry out any computation that can be formulated in numerical terms[59], implying a memory and a processor, whereas “in a special-purpose machine these instructions are an

59 Another term often associated with defining computing is algorithm, which itself has a rich etymology I will not review here. B. Jack Copeland defines algorithm as “a 'mechanical' or 'moronic' procedure for achieving a specified result. That is to say, an algorithm is a finite list of machine-executable instructions such that anyone or anything that correctly follows the instructions in the specified order is certain to achieve the result in question” (337).

integral part of the device and constitute a part of its design structure” (1). The processor implements control to autonomously and automatically execute orders stored in memory like a reader to a book or player piano to scroll.[60] That they are fully automatic, independent of any human operator after the computation starts, is the key characteristic ascribed by contemporary theorists like Kitchin and Dodge. Another entrenched feature of everyday electronic computers concretized in the design of the EDVAC hinged on their key decision to use a binary number system instead of the traditional decimal system. It is a well-argued design affordance that nonetheless requires an alien perspective to use, though they also think future technologists will train themselves appropriately. “We feel, however, that the base 10 may not even be a permanent feature in a scientific instrument and consequently will probably attempt to train ourselves to use numbers base 2 or 8 or 16” (8).[61]

The aspect of electronic computing most crucial to Kittler, at least in *Gramophone Film Typewriter*, is also introduced in this document: conditional and unconditional transfers as basic control structures beyond sequential processing. “The utility of an automatic computer lies in the possibility of using a given sequence of instructions repeatedly, the number of times it is iterated being either preassigned or dependent upon the results of the computation. . . . We introduce an order (the conditional transfer order) which will, depending on the sign of a given number, cause the proper one of two routines to be executed. . . . This unconditional transfer can be achieved by

60 Floridi provides an excellent logician's analysis of the stored program computer, as does Weizenbaum. Clearly, developing a 'tutor text' for teaching the stored program concept is essential to any philosopher of computing.

61 The binary system is often contrasted Babbage's decimal machinery, which for its own part introduced the strange system of differential arithmetic retained by these newer binary devices.

the artificial use of a conditional transfer or by the introduction of an explicit order for such a transfer" (3).

Input and output organs permitting machine signaling to and manipulation by humans form the final constituent of the stored program computer. Where contemporary computing machinery differs the most from the ancestor described here, interactivity between machine and human was limited to typewriter input for ad hoc data input, and a single machine instruction deployed to halt computer and notify completion by flashing a light or ringing a bell. It is safe to claim that this document and others like it from the early days of electronic computing open an enormous, unexplored territory for future research projects uncovering, in a sense, what we already know, or have forgotten having given this knowledge over to the machines that manufacture new machines. Despite the common ancestry of the von Neumann architecture, Manovich points out that "the most important example of such non-deterministic development is the invention of the modern interactive graphical human-computer interface itself by Sutherland, Engelbart, Kay and others. None of the key theoretical concepts of modern computing as developed by Turing and Von Neumann called for an interactive interface" (*Software Takes Command* 97).

Histories of computing come in various forms: some attempt to cover all of human history from the stone age to the present day, others trace the origins of the electronic era from World War II onward, still others the origins of the personal computer, or some focus on the development of a particular model like the Apple Macintosh. Michael R. Williams' *A History of Computing Technology* and Paul E. Ceruzzi's *A History of Modern Computing* exemplify the first and second categories, respectively, and are the work of professional historians associated with the IEEE Computer Society. The personal computer appears in many popular press books like Steven Levy's *Hackers: Heroes of the Computer Revolution* and Paul Freiberger and Michael Swaine's *Fire in the Valley: The Making of the Personal Computer*. Levy's later publication *Insanely Great* offers a history of the Apple Macintosh.

Martin Campbell-Kelly and William Aspray's *Computer: A History of the Information Machine* strikes a middle ground while offering to paint a picture that explains their social construction, transforming from roles in secret projects used by small circles of scientists and atomic weapons designers, to everyday word processing and business record keeping, arriving at the personal computer and the Internet. In their preface they claim that "the aim of the series is to convey both the technical and human dimensions of the subject: the invention and effort entailed in devising the technologies and the comforts and stresses they have introduced into contemporary life" (vii). Books of this genre, serious attempts at presenting minimally biased histories of the evolution of state of the art through multiperspectival contingencies, form the foundation of philosophy of computing. They predict deeper understanding of computers than their broad definition of information machines through emergence of synthetic historical scholarship of the sort I am trying to lay out in this dissertation. "Our work falls in the present generation of scholarship based on the broader definition of the information machine, with strong business and other contextual factors considered in addition to technical factors. We anticipate that within the next decade, a new body of historical scholarship will appear that will enable someone to write a new synthetic account that will deepen our understanding of computers in relation to consumers, gender, labor, and other social and cultural issues" (6).

I follow Spinuzzi, however, in advocating the need for a syncretism rather than a synthesis. Suggesting “we need a more flexible, more associational activity theory with a stronger splicing account if we're going to analyze net work properly. We need activity theory to be dialogical and more rhetorical,” (206) he does not wish to create yet another theoretical framework but rather deploy simultaneous, multi-perspectival methodologies. Similarly, as I build this heterogeneous framework of disciplines I mean to use them all together to engage the collective intelligence problems laid out as the as-is situation.

Discussing the early years of personal computers, Campbell-Kelly and Aspray give an example of a relevant but overlooked influence on the development of the as-is situation. “Computer games are often overlooked in discussions of the personal-computer software industry, but they played an important role in its early development. Programming computer games created a corps of young programmers who were very sensitive to what we now call human/computer interaction. The most successful games were ones that needed no manuals and gave instant feedback” (250). Indeed, user-friendliness via graphical user interfaces was the next step in broadening computer use far beyond Kemeny's vision of BASIC programming on time-sharing systems. “For the personal computer to become more widely accepted and reach a broader market, it had to become more user-friendly. During the 1980s user-friendliness was achieved for one-tenth of computer users by using a Macintosh computer; the other nine-tenths could achieve it through Microsoft Windows software” (264). The minority market share held by Apple resulted from its marketing strategy of product differentiation, going all GUI with proprietary hardware and operating system software where Microsoft played to the generic IBM PC clone market. We could ponder whether this product differentiation marketing strategy by

Apple may have contributed to the shift toward postmodern interface-level preferences Turkle articulates in *Life on the Screen*.

Histories of software often overlap histories of computers, no doubt because software emerged from hardware as stored programs became easier to manipulate and programming languages evolved.[62] As Edwards points out, the earliest programmers were mathematicians and engineers who were close to the hardware. Higher-level languages with interpreters and compilers were needed for nonexperts, requiring more memory and machine time to support their execution. Short Code was the first interpreted assembly language, created by Eckert and Mauchly for the 1949 BINAC. "Employing alphanumeric equivalents for binary instructions, Short Code constituted an assembly language that allowed programmers to write their instructions in a form somewhat more congenial to human understanding. . . . A separate machine-language program called an interpreter translated Short Code programs into machine language, one line at a time. . . . But instructions still had to be entered in the exact form and order in which the machine would execute them, which frequently was not the conventional form or order for composing algebraic statements" (247). Indeed, compilers were originally called *automatic programmers* when the Office of Naval Research sponsored symposia in the mid 1950s. "Programming still referred to the composition of the machine-language instruction list. Algorithms written in a higher-level language were not yet seen as actual programs, but rather as directions to the compiler to compose a program; compilers thus performed automatic

62 A section on histories of programming languages will be reserved for future research, with only a few notes made here; see "Philosophical Programmers," where the Knuth Pardo text on early programming languages provides much denser coverage than Edwards' comments.

programming. The independence of symbolic levels in computing had not yet achieved the axiomatic status it later required” (249).

Edward's text provides interesting historical connections between the development of computer technology and the Cold War mentality he calls the *closed world*, exemplified by the image of the world enclosed by radar grids, threatened at every point by long range missiles, as the two great superpowers proliferated ever stronger deterrents against each other: the military-industrial-university triangle. He is careful to explain the roles of the various actors as loosely coupled rather than deliberately planned. “Academic psychologists and computer scientists generally did not understand the major role they played in orienting the military toward automation, symbiosis, and artificial intelligence as practical solutions to military problems. . . . They could do so precisely because for the most part there was no scheme, in the sense of some deliberate plan or overarching vision. Instead, this larger pattern took the form of a discourse, a heterogeneous, variously linked ensemble of metaphors, practices, institutions, and technologies, elaborated over time according to an internal logic and organized around the Foucaultian support of the electronic digital computer” (272).

This point is crucial for recognizing the difference between treating the history of computing as a monolithic, factual retelling of key events leading to the current state of the art, as appears in popular press books like *Hackers* and *Fire in the Valley,* and treating it as many sets of contested narratives that mean different things to different people. Once again I am gesturing toward the need to interrogate an implicit ontological binarism with respect to collective intelligence. The theoretical framework of procedural rhetorics of diachrony in synchrony is my attempt to articulate a position between the poles of Foucault's dust and global conspiracy.

Campbell-Kelly claims his 2003 book *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry* “is perhaps the first attempt at writing a full-length history of the software industry broader than the study of an individual firm, heavy use of corporate archives was not appropriate, nor indeed was it possible. Instead, it is based largely on monographic studies of the software industry, the periodical literature, and reports by industry analysts” (22-23). While Campbell-Kelly intends this statement, and the rationale for his study, to appeal to the fact that ninety percent of publications about the history of software deal with Microsoft and Bill Gates, while Microsoft software only accounts for ten percent of the overall software industry, he also appreciates the importance of utilizing a broad spectrum of sources for his research methodology. As Black pointed out concerning his research into IBM's involvement in the holocaust, it was difficult if not impossible to survey corporate records due to their being classified as sensitive information, poorly indexed, stored in vast paper archives, or simply lost; moreover, he had to deal with intentional obfuscation “employing deliberate vagueness, code words, catchphrases, or transient corporate shorthand. I had to learn the contemporaneous lexicon of the company to decipher their content” (13). Thus we tend to build retrospective explanations of imagined intentions and plans that yielded the public phenomena under study. The most plausible accounts become the historical narrative that inevitably leads to precisely the outcome that seems to have so fortuitously materialized among a range of possible, just as likely outcomes.

In the case of software, its ontic status became an open question once intellectual property protections were sought. “The law offered several forms of intellectual property protection: patents, copyright, trade secrets, and trademarks. . . . Exactly what kind of an artifact software represented was an open question. . . . Informatics decided to rely on trade secret law for protection. It required its customers and its employees to sign non-disclosure agreements" (107). Later, software would be protected by the 1976 Copyright Act. Thus, despite the proliferation of texts written about Microsoft and Bill Gates, it is largely from secondary sources. He gives the exception of Autodesk's publication of its early strategic thinking in *The Autodesk File*, "but until Microsoft opens its archives to independent scholars, we have only some tantalizing hints of the company's strategic thinking and the extent to which Gates was responsible for it" (243).

The emergence of free, open source software has fundamentally altered the availability of 'corporate archives' to investigators. Online foss communities provide historically unprecedented access to the equivalent of corporate archives of a very large number of software projects. Campbell-Kelly predicts free, open source software will be next important scholarly topic beyond his book cutoff of 1995, but notes "I don't know what it is, but I bet there is something much more important going on right now than Java, Linux, or open-source software, and that it will be 2010 before it becomes fully apparent" (10). Ironically, an assessment of that outcome from the vantage point of 2015 might be that the enormous market for mobile applications for Apple and Android devices has ushered in a new era of proprietary, closed source software. The tide may be turning against open protocols as well because each app conceals its communications in encrypted data streams.

Another computing and software history genre consists of biopic interviews of people who are credited with inventing or promoting systems and applications that are deemed extremely successful commercially or indispensable keystones in the chosen historical progression. One example is Dennis Shasha and Cathy Lazere's *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists*. They comment that “in most sciences, the seminal thinkers lived in the remote past. To uncover what they did and why they did it, we must scavenge in the historical record, picking among scraps of information, trying to separate facts from mythology. . . . Computer science is different. The mathematicians who first studied computation in its current form—Alan Turing, Emil Post, and Alonzo Church—did their work in the 1930s and 1940s. Their conception of the computer is one we still live with: a calculating engine and a memory for storing instructions as well as data” (ix). More to the point, a large number of these influential computer scientists, programmers, and entrepreneurs are still alive working and teaching. Thus when it comes to histories of computer networking—telling the story of the Internet has taken the place of the personal computer as the obligatory creation myth included in countless introductory texts—techniques include interviews of the individuals who were involved.

The value of this approach is two-fold. First, the opportunity is given for development of heterogeneous narratives, especially in the sense recommended by Callon to let the engineers speak for themselves. Second, and more specific to the aims of my project, the material that comes to print from such interviews and primary sources retains the minimally interpreted words and handiwork of the innovators themselves. As our theoretical frameworks evolve, these texts can be visited again and again to corroborate hypotheses and suggest whole new avenues of philosophical approach. Thus I read through *Out of Their Minds* and Susan Lammers' *Programmers at Work* seeking hints about personal programming styles. The fact that some of these texts, like Lammers, includes code snippets and diagrams allows the investigation to go beyond human-to-human rhetoric, into the unexplored territory where human and machine cognition converge. In the upcoming presentation of my methodology I call this territory *working code places*.

Members of innovation communities may take the initiative to directly engage with scholars. Such is the case with Katie Hafner and Matthew Lyon's *Where Wizards Stay Up Late: The Origins of the Internet*. In the Acknowledgments at the end of the book they stress the importance for scholars of history of software and technology-advancing organizations to have access to archives, funding, and assistance from such organization's librarians. “This book grew out of an idea that originated with engineers at Bolt Beranek and Newman. Memories were growing fuzzy in late 1993, when we first started thinking about doing a book, and Frank Heart and others were interested in having BBN's considerable role in the creation of the original ARPANET recorded. Not only did the company open its archives to us and cooperate in every way but it helped fund the project as well, while agreeing to exercise no control over the content of the book. Marian Bremer, then BBN's head librarian, made the initial phone call that led to the book” (287). That BBN even had a lead librarian, who took the initiative that led to the writing of the book, confirms the point made by Campbell-Kelly and others at how slim the chances are of capturing much of that early history as remains forgotten in archives or has been destroyed.

It is instructive to read Janet Abbate's *Inventing the Internet* alongside Hafner and Lyon. She claims to apply a unique SCOT approach to computer communication, involving narratives of origins, production and use. “Relatively few authors have looked at the social shaping of computer communications. There have been many social and cultural studies of computing in recent years, including compelling analyses of networking by Sherry Turkle, Gene Rochlin, and

Philip Agre, but these works tend not to examine in detail the origins of computer technologies, focusing instead on how they are used once they exist" (4). The practice and meaning of computing was redefined by Internet long distance interaction, as Manovich argues the emergence of personal computers led to cultural software. Indeed, her interest in the Internet grew from her experience of it as a curious alliance of military and civilian interests. "My curiosity about the Internet grew out of my experiences as a computer programmer in the mid 1980s, when few people outside the field of computer science had heard of this network. I was aware that the Internet had been built and funded by the Department of Defense, yet here I was using the system to chat with my friends and to swap recipes with strangers rather like taking a tank for a joyride!" (2).

Abbate contends that predictions by APRA about users' benefits were wrong, so its success needs to be explained rather than taken for granted, focusing on active users who played a role in its development. We need to go beyond simple view of consumers amassing after a technology has been delivered to the market. Using the early ARPANET was challenging, contrary to the ease of access and use taken for granted today with the Internet. It was transformed by virtual communities built by experiments building features like sharing information, support, and recreation in the network environment. Reliance on local knowledge led to development of online documentation, system announcements, email support that spread virtual community and became our habitual practices of interactivity. At the same time, unexpected advantages of networked computing were realized by giving researchers access to other machines, programming languages, and services. "The ARPANET changed the way computer scientists worked and the types of projects that were feasible" (100).

Information sharing and enhanced collaboration were noted by computer scientists as key benefits of networking, transforming how science was done. Nonetheless, users did not collaborate like the model promoted by SRI's NIC, using mostly email and file transfer due to their simplicity compared to the NLS developed by Engelbart. Email use and participation in virtual communities altered common perception of ARPANET as communications system more than computing system (111). So computing began its descent into the background as internalized interface use, and the evolved behavior Nigel Thrift attributes to coaction of technical unconscious with human endeavor. I next turn to psycho-social studies of computer programmers to provide a perspective absent from analyses that focus on the consumption of these computational artifacts.

Psycho-Social Studies of Computer Programmers

The familiar sequence of tasks for programming projects, which became known as the waterfall model, arose in a 1956 lecture by John F. Jacobs at the Lincoln Labs and diffused throughout the software industry from the SAGE project programmer exodus (Campbell-Kelly 67-69). It should be topic in the philosophy of computing for its reflection of social and cognitive norms, social construction, as well as likely reflexive relationships to the evolution of technological systems along with human thinking. Studies of the psychology, sociology, learning, and ethnography of computer programmers and software development communities therefore present themselves as another component informing my theoretical framework. The best known representative of this group is Frederick P. Brooks, Jr., who led the development of IBM's *System/360* Operating System effort, and then went into teaching computer science. His criticism of the 'mythical man-month' and the fallacious thinking of software development managers regarding software project scheduling is as pertinent today as it was forty years ago when his famous book by the same name was published. He contends that the success of software projects is largely impacted by lack of calendar time due to poor estimating, confusing effort with progress, poor monitoring, and finally adding manpower when slippage has been noticed.

The root cause is fallacious reasoning. First, there is pervasive optimism about programming. “In many creative activities the medium of execution is intractable. Implementation, then, takes time and sweat both because of the physical media and because of the inadequacies of the underlying ideas. The programmer builds from pure thought-stuff: concepts and very flexible representations thereof. Because the medium is tractable, we expect few difficulties in implementation; hence our pervasive optimism. Because our ideas are faulty, we have bugs; hence our optimism is unjustified” (15). Second, there is pervasive misunderstanding about the relationship between effort and progress. “The second fallacious thought mode is expressed in the very unit of effort used in estimating and scheduling: the man-month. Cost does indeed vary as the product of the number of men and the number of months. Progress does not. *Hence the man-month as a unit for measuring the size of a job is a dangerous and deceptive myth*. It implies that men and months are interchangeable” (16).

People and months are only interchangeable when tasks can be partitioned with no communication needs. Communication adds the burden of training and intercommunication, quickly dominating task time as new workers are added. Sequential constraints also prevent task partitioning, where another famous Brooks quote appears: “The bearing of a child takes nine months, no matter how many women are assigned. Many software tasks have this characteristic because of the sequential nature of debugging" (17). The common practice of scheduling to match a date desired by a patron commonly leads to shortened testing periods, for testing is usually the most misleading part of schedule because of optimistic expectation of less bugs. Brooks claims he used one third planning, one sixth coding, one quarter component test, one quarter system test as his scheduling rule of thumb, although few of the conventionally scheduled projects he studied allowed one half of the time for testing. Thus the waterfall model that flowed from military project management into commercial industry carried with it its own inefficiencies.[63]

Everyday experience using software shrouds these vicissitudes of software development behind a finished product, although traces are left in delayed releases, inconsistent interfaces and controls, and nagging bugs. To make a psychological observation of my own, I suggest that most people who do have some experience programming got it working alone on hobbies or in classroom settings using personal computers. According to Campbell-Kelly, the established computer industry did not perceive microprocessor-based stand-alone devices as a threat because they were developed in the electronics industry. Thus microcomputer software development practices reflect bricoleur, dilettante origins, as suggested by examining the emergence of the Altair 8800 and folk history of the personal computer. "Although some professional software development practices diffused into microprocessor programming, much of the software

63 Indeed, forty years later these comments are still appropriate for the mid size software development firm where I work. Various other programming management methodologies have been developed and tried throughout the years, but to date (as Brooks reaffirms in the 20th anniversary edition of *Mythical Man Month*), there is still no silver bullet. Rosenberg's research presented in *Dreaming in Code* investigates whether the open source development model provides any better results.

technology was cobbled together or re-invented, an amateurish legacy that the personal computer software industry took several years to shake off. . . . The launch of these genuine consumer products created a significant consumer market for personal computer software" (203). He goes on to note the 'bedroom coder' phenomenon that was fueled by popular magazines combined with the lack of significant barriers. "No additional software development system was needed, there were no proprietary trade secrets to unlock, and programs could be duplicated on the computer itself, with no need for access to a third-party manufacturing plant" (276-277). Thus a contrasting perception of software development arose in popular culture compared to the tar pit Brooks portrays. Yet it does not take much searching to find a substantial body of psycho-social research dating back to the late 1960s and early 1970s concerning computer programmers and their workplaces deeply influenced by Brooks's lead. This brings us back to Gerald Weinberg's 1971 *Psychology of Computer Programming*, whose ironic naiveté concerning IBM's secretive support of Nazi punch card machinery I quoted in the first chapter.

Weinberg's overall research question is how we can study programming. He sought to initiate a new field investigating computer programming as a human activity, the psychology of programming, as I likewise wish to articulate critical programming as a new digital humanities practice that takes its cue, in part, by playing catch up with the an extensive body of existing literature that has not been widely read by humanities theorists presented in this and the previous sections. Campbell-Kelly and Aspray declare that a new type of rhetoric is required for computer programming. "The fundamental difficulty of writing software was that, until computers arrived, human beings had never before had to prepare detailed instructions for an automaton—a machine that obeyed unerringly the commands given to it, and for which every possible outcome had to be anticipated by the programmer” (181). Later in the section on Software Studies, I will introduce Bogost's take on it as *procedural rhetoric*, whose quintessential form is programming activity but quickly diffuses into game play and user interface manipulation in general.

Arising from computer science, the psychology of programming exemplifies one end of a continuum whose humanities end is still forming. Indeed, Weinberg presumes that “it does not seem advisable to give such a course to people who are not able to write programs themselves, or who are at lower than a graduate level or a senior level with a strong major in Computer Systems or Computer Science” (ix), setting a high bar for entrance. Programming as a kind of writing connects the period to late literacy before born digital generations, with the twist that the capability of directly recording user behavior alters this human computer relationship from other activities that can be studied. Moreover, the advent of terminals connected to time-sharing systems, and later of personal computers, seems to have consigned the practice of reading programs as a whole obsolete. “Just as television has turned the heads of the young from the old-fashioned joys of book reading, so have terminals and generally improved turnaround made the reading of programs the mark of a hopelessly old-fashioned programmer” (6). Programmers have become accustomed to working on small units of code at a time on account of their low resolution text displays, whereas back in the days of batch processing, programs were read from long paper printouts, annotated, and thought over, before the next appointment with the card punch or teletype machine to painstakingly recode them.

Yet reading code is exactly what Weinberg initially recommends. “Perhaps if we want to understand *how* programmers program to lift the veil of the programming mystique we could fruitfully begin by seeing what is to be learned from the reading of programs” (6). He suggests we read programs by asking why each part exists rather than making a serial traversal from beginning to end like reading a novel (6-7). We quickly find, however, that comments are rarely left concerning overcoming machine limitations such as handling precision, real numbers and intermediate storage, even though much programming effort is expended to solve such problems (8). Comments may even hinder debugging efforts. “The whole idea of a comment is to prepare the mind of the reader for a proper interpretation of the instruction or statement to which it is appended. If the code to which the comment refers is correct, the comment could be useful in this way; but if it happens to be incorrect, the set which the comment lends will only make it less likely that the error will be detected” (164).

Numerous experiments demonstrate that psychological set and distance can be impediments to error location activities. Syntax highlighting and other visual cues of the interface help counteract by offering machine responses. “One of the first lessons the novice programmer learns is to make careful distinction between his handwritten zero and oh, if someone else is keying his programs” (162). It was less likely even in Weinberg's day that a helper keys in programs written on paper; however, he notes increasing carelessness in choosing symbols as a side effect of successful automatic error detection. Mnemonic symbols induce torpor by those reading program code, further intensifying effects of psychological set.

A key to intelligent behavior involves flexibility manipulating assumptions and applying the appropriate formulas to solve problems. Memory helps too, especially “by enabling him to work on problems when he does not have all his papers in front of him” (167). While all of these facets of reading code may be shared by professional and bedroom coders alike, other aspects of the psychology of programming are unique to professional, group settings. Therefore, one purpose for examining psycho-social studies of computer programmers and programming communities is to adjust prejudices about the social practice of programming based on individual educational and hobbyist experiences.[64]

The heart of Weinberg's research and teaching about the psychology of computer programming pertains to professional environments. Specifications evolve during development. Writing programs is a learning process for all parties involved. “In most cases, we do not *know* what we want to do until we have taken a flying leap at programming it. . . . Specifications evolve together with programs and programmers. Writing a program is a process of *learning* both for the programmer and the person who commissions the program” (12). Contrary to Kittler's solipsistic remark that anyone who has written a line of code knows the fruitlessness of trying to describe to another person how they solved a particular problem, Weinberg contends that meanings proliferate. “There is a difference between a program written for one user and a piece of software. When there are multiple users, there are multiple specifications. When there are multiple specifications, there are multiple definitions of when the program is working” (19).

This necessary transcendence of the individual in professional settings imbricates social conflict in addition to other communication challenges. Programs do not seem to be written for reuse even though programmers know code continues to be used when they are not involved with it. Fisher's adaptability theory explains the surprising difficulty of adapting well working systems: “the better adapted a system is to a particular environment, the less adaptable it is to

64 Nonetheless, much computer science and management information systems pedagogy involves group projects and team work, and participation in foss communities gives a sense of what professional environments are like.

new environments" (21). Therefore, coding poses a challenge for both efficiency and adaptability despite bias of both in the rhetoric of high level, object oriented programming paradigms.[65]

Beyond these psychological factors that affect the design and literal content of program code, the history of program development leaves traces beyond machine, language, and programmer limitations, especially due to the size and composition of original programming groups. "Even the very structure of the program may be determined by the size and composition of the programming group that originally wrote it since the work had to be divided up among a certain number of people, each of whom had certain strengths and weaknesses" (11). We might compare the design of large programming projects to the sedimented composition of psychoanalytic unconscious, although the insights of SCOT theorists have already provided a language appropriate to technological artifacts.

This elasticity needed to work with rigid machines counters folk wisdom that programmers should resemble their medium. The idea of egoless programming that Weinberg touts as the missing ingredient managing effective teams is training people to accept their inability to function like a machine to always produce perfect work. Programming requires learning to practice a new type of rhetoric. Weinberg's account of egoless programming suggests advantages beyond detecting errors from the sense of always writing code for future readers.

65 Weinberg concludes this otherwise correct maxim of common sense computer programming folk psychology with a sexist cast that is embarrassing to read today: "Asking for efficiency and adaptability in the same program is like asking for a beautiful and modest wife" (22). There are numerous references to women in his book that challenge the patience of today's readers. It begs the question to ask what might we encounter if we got our hands on the suppressed IBM publication purportedly glorifying the punch card machine configuration feats of Nazi engineers.

Egoless programming practices would likely help meet specifications, scheduling, estimating, and continuity, four key factors of good programming. However, he contends that even if it was widespread—and he notes default practice is isolated individual programmers relying on their personal competence, and in competition with each other for prestige—the practice of egoless programming is likely concealed behind proprietary business practices and smug satisfaction of happy workers, as well as lacking tests of strength in sense of Latour, Boltanksi and Chiapello. It seems like a positive sense of flexibility compared to that required by the new spirit of capitalism to adapt to flexible environments decried by Boltanksi and Chiapello.

Nonetheless, the relationships between programmers and their managers looms large for a large portion of the book before turning to psychological studies related to the tools of the trade. Weinberg discusses the second NATO conference on Software Engineering held in 1969 as a watershed moment in the history of the software industry, quoting Joel Aron. “We analyzed the reasons for failure, as given to us by the project managers, and by people who had performed previous evaluations of the projects. They gave various reasons behind the failure of the projects, *virtually all of which were essentially management failures*. We ran into problems because we didn't know how to manage what we had, not because we lacked the techniques themselves” (112). To respond to this productivity problem, Philip Kraft opens *Programmers and Managers: The Routinization of Computer Programming in the United States* by describing an identity crisis confronting computer programmers: are they are managers or engineers?

He casts the problem by referring to Weiner's understanding “the intimate and delicate relationship between control and communication: that messages intended as commands do not necessarily differ from those intended simply as facts” (v). As an unintended consequence of the stored program architecture that has been part of mainstream computing since Burks, Goldstine and von Neumann, the ambiguity between control and content seems present at the human level of programmers, too. “Are they primarily members of management acting as foremen, whose task it is to ensure that orders emanating from executive suites are faithfully translated into comprehensible messages? Or are they perhaps simply engineers preoccupied with the technical difficulties of relating software to hardware and vice versa? Are they aware, furthermore, of the degree to which their work whether as manager or engineer routinizes the work of others and thereby helps shape the structure of social class relationships?” (vi).[66]

Surveying existing literature on programmers, Kraft finds most of it written by managers with the concern of imposing discipline rather than writing better programs; they fall into the categories of moral uplift, psychological profiling, industry statistics, and ethnographies. Manager interviews revealed programming work being organized like any other work in corporate bureaucracy. Only the last category, where he includes F.T. Baker, Harlan Mills, and Gerald M. Weinberg, addresses the importance of political, social relations of the workplace, highlighting power, domination, subordination.

Programmers seemed unaware of organizational processes. The programmer/manager relationship was often viewed as personal rather than organizational, with surprising compliance to the manager's opinion. Kraft's overall finding—or barely couched political agenda—accuses the workplace deployment of structured programming as the quintessential managerial selection process intended to de-skill and control workers, aims that are not inherent in the technology

66 Kraft's insightful connection between technological forms and social forms anticipates those of Catherine Malabou, who argues that contemporary neuronal ideology mirrors the political and social functioning in flexible forms of capitalism.

itself as articulated by Edsger Dijkstra and David Gries (9). “In spite of the lack of clear-cut boundaries between the software suboccupations, the divisions point to what may be the most important aspect of modern programming: the work of head and hand have been separated and given over to difference people” (16).

As the final users of computing machinery were large organizations, they were seen by them primarily as labor saving machinery. Several new, specialized occupations arose to transfer operation and maintenance away from their end users. “The desire of organizational users to treat computers like other labor-saving machinery intensified the already great pressure on hardware makers to simplify the computer's use. The goal, expressed early and often in the short history of the computer, is the same as in all other engineering industries: managerial control. . . . [quoting Robert Bosak] The ultimate is to remove the programmer entirely from the process of writing operational programs" (27). Separating the head from the hands meant producing programs was left to an anonymous army who have little understanding of why they are doing what they do. “At least up to now, the computer has intensified, not reduced, the separation between those who think and those who do everything else, an division now beginning to separate software workers as well” (29).

To Kraft de-skilling “is a deliberate effort to transform work made up of separate but interdependent tasks into a larger number of simpler, routine, and unrelated tasks” (51-52). De-skilling is standardizing work to produce standardized products, and “in effect, the skills and talents of a relatively tiny proportion of the labor force—research scientists, production engineers, systems analysts, and similar creative workers—have been used to make unproductive the skills of a vast majority of the rest” (52). Observation of three workplaces, interviews, and published sources ground this workplace analysis (66-67). His conclusion is that the extensive use of canned programs, structured programming, and Chief Programmer Teams are central changes making professional programming less complex and more routine. “The fragmentation of programming into suboccupations like coding, program design, and systems analysis suggests that the industry has been able to introduce changes in the way programming is done to render it less complex and more routine” (54). Structured programming and modularization encourage programmers to work like machines. “If there are only a few pre-determined ways of ordering a program's logic, considerably less skill, training, and experience are required to grasp the major logical tools of the trade. Moreover, since the procedures are few, universal, and well-understood, one programming routine will develop much like all others. . . . Both writing and debugging are further facilitated by structured programming's logical partner, modularization, which breaks down an entire software system into limited-function, discrete units, written independently and then fitted together in a pre-determined way to form a single system” (57).

The unasked question relevant to my study is whether autonomous technologies transfer such de-skilling performed by previous generations to the present one, and how collective intelligence has been affected. Since structured programming has been largely been supplanted by object-oriented programming as the fundamental rhetoric of writing for machines, and the model for canned routines has been extended to accommodate not just in house and commercial libraries but also the components from the free, open source marketplace, a new set of analyses are required, and they abound in technical journals and publishers targeting the IT industry. Just as Boltanski and Chiapello analyzed management texts to derive the contours of the new spirit of capitalism, a similar procedure could be employed to discern the new spirit of computing. I will reserve the presentation of a few of these contemporary studies as methodological examples of working code places later in the chapter.[67]

While I justified examining psycho-social studies of computer programmers and programming communities for adjusting prejudices about the social practice of programming stemming from individual educational and hobbyist experiences, it is important nonetheless to consider how children become acculturated to working code, and how programming is taught and learned at the informal level corresponding to Kittler's naturalization of literacy via the Mother's Mouth, where “phonetic reading instruction is thus a writing system, and not merely a method of speech. Only as such, through the regulation of pronunciation according to a hypothetically 'accepted' norm, could it teach children orthography prior to any instruction in penmanship” (*Discourse Networks 1800/1900* 35-37).

Sherry Turkle, whose interest in the psychological effects of computer technologies led her to study children using them in various settings, was surprised to find four-year-olds writing simple programs in the early 1980s. “Computers bring writing within the scope of what very young children can do. It is far easier to press keys on a keyboard than to control a pencil. . . . The computer has become the new cultural symbol of the things that Rousseau feared from the pen: loss of direct contact with other people, the construction of a private world, a flight from real things to their representations. . . . If our ideas about childhood are called into question by child writers, what of child programmers? If childhood innocence is eroded by writing, how much more so by programming?” (*Second Self* 94-95). While her later work clearly shifts from

67 There are numerous ethnographies and collections of empirical research that I have to leave on the table by Ensmenger, Feller et al., MacKenzie, Mayer, Rosenberg, and others included in the List of References.

any emphasis on programming to everyday application use, the triumph of the surface over deep structure that led her to declare the computer the quintessential postmodern object, this period during which it seemed inevitable that programming would become naturalized like handwriting and phonetic reading provides a rich ground from which to theorize the trajectory that brought us to the dumbest generation.

The programming languages used by the children she studied included BASIC, PILOT, and Logo. Graphics were a favorite area for children learning to program, and she noted a considerable amount of sharing. “Children can't do much with each other's book reports, but they can do a great deal with each other's programs. Another child's program can be changed, new features can be added, it can be personalized” (100). What piqued her interest in these ethnographic studies were the cultural extremes represented by programming styles regarding comfortable manipulation of formal objects versus impressionistic development of ideas relying on language and visual images. “Some children [Jeff] are at home with the manipulation of formal objects, while others [Kevin] develop their ideas more impressionistically, with language or visual images, with attention to such hard-to-formalize aspects of the world as feeling, color, sound, and personal rapport. Scientific and technical fields are usually seen as the natural home for people like Jeff; the arts and humanities seem to belong to the Kevins” (104).

Noticing these patterns led her to propose two fundamental styles or approaches children take toward programming: hard and soft mastery. Hard mastery implements plans to impose engineering and scientific will over the machine; soft mastery prefers emergent, iterative, artistic interaction with the media. She invokes the distinction between the scientist and the *bricoleur* by Claude Levi-Strauss: “while the hard master thinks in terms of global abstractions, the soft master works on a problem by arranging and rearranging these elements, working through new combinations: (105). Computers allows bricoleurs to operate in the formal domain, although in her later work Turkle argues that the surface reigns as technology evolves. At the same time, the distinction echoes a cultural division by gender between hard mastery for boys, soft mastery for girls, which is expected to be repeated in the world of things as it is in the world of people.[68]

Crucially, Turkle argues that programming style expresses personality, not just reflects a specific computer architecture imposed on programmer; "looking closely at Jeff and Kevin makes it apparent that a style of dealing with the computer is of a piece with other things about the person—his or her way of facing the world, of coping with problems, of defending against what is felt as dangerous*" (105). Nonetheless, she provides examples of how programming enhances other means of learning, and how computer use projects internal experience, like a symptom. It also becomes a basis of belief formation about people, a topic consistently carried forward to her recent publication *Alone Together*.[69] Her overall conclusions in the final chapter of *The Second Self* are that the computer provides a new window onto developmental processes, a projective screen for personality styles as well as actually entering into cognitive and emotional development, and a medium for growth and getting stuck.

68 Consider how new configurations between closed and open source, for example, may be used to explore other relations beyond gender and science. My preliminary research of philosophical programmers suggests a multiplicity of programming styles that explodes Turkle's hard and soft mastery dichotomy.

69 Her latest book *Reclaiming Conversation: The Power of Talk in the Digital Age* builds on this book's conclusion that w are at the center of a perfect storm in the present technological milieu, and need to to pursue reclaiming good manners, privacy, and concentration.

Going back to the asterisk at the end of the previous quotation points to a brief footnote dismissing the specificity of different platforms and languages as relevant to revealing individual programming styles. “Not all computer systems, not all computer languages offer a material that is flexible enough for differences in style to be expressed. A language such as BASIC does not make it easy to achieve successful results through a variety of programming styles” (105). In another footnote in a later chapter, she acknowledges the appropriateness of various languages for various tasks following that statement that computer cultures are built from preferences in programming languages and styles (179). She still does not reflect the other way around, how using particular languages and platforms may affect the preference formation of coders. She does not dive into questions of how *particular* machines affect development of personal styles of working with the machines.

Turkle's position forecloses a number of approaches that are relevant to my work in this dissertation: what *other* programming styles beyond hard mastery and bricolage might be discerned? What stylistic preferences do various architectures and languages reflect, or encourage? Do children gravitate toward particular platforms, languages, and programming styles on account of their pre-existing personalities, or might their development be influenced by the simple fact, for example, that their parents purchased a particular brand of machine for them, their school had those systems in their classroom, or their 'Computer Lit' instructors happened to be familiar with them? To contemplate these questions requires us to leave psycho-social studies and engage at a more basic level that I am calling philosophies of computing technologies, and then narrow our focus to software, code, and finally the specific platforms on which all software, code, and network phenomena subsist.

Philosophers of Computing Technologies

Echoing the sentiment of Bush, Goldstine, and von Neumann, it would take me too far afield to provide a substantial section on the philosophy of technology, although my research included much reading from this discipline.[70] It is hard to draw a line between thinkers providing sustained philosophical reasoning for their interpretive approach to technology, and those employing historical scholarship and reasoning that includes substantial examples from computing technologies to support a broader philosophical agenda. This section focuses on three thinkers, Josepth Weizenbaum, David Golumbia, and Alexander Galloway, whose broad philosophical musings about computing technologies recapitulate much of the reasoning that led to my conception of the as-is situation of the post-postmodern network dividual cyborg, whereas the two following sections foreground theorists who specifically align themselves with software studies and code studies.

A crucial movement performed by many philosophers of computing technologies is to shift the notion of code from abstract, immaterial characterizations to ones appreciating fundamental materialities that should not be elided. Compare it to Suchman's plans and situated actions: the abstract notion of computing outlined by Turing and carried forward by Floridi and Copeland describe disembodied logical systems for which all possible permutations of expressions reduce to elegant mathematical proofs, retrospectively described like plans. For any specific machine architecture, there are non-trivial conventions that must be followed in order to write programs that actually work, countless situated actions. Thus we ought to consider John

70 Philosophers of technology I encountered in my early research include Heidegger, McLuhan, Foucault, Heilbroner, Mitcham, Ihde, and many others from the anthology by Scharff and Dusek.

Uffenbeck's unique perspective on stored program computing that considers it in terms of four basic fetch and execute instruction cycles constituting all complex microprocessor instructions, “to consider the effects each computer instruction has on the electrical lines or buses of the microprocessor chip itself. . . . The instruction set of a computer can be thought of as a list of commands that cause unique sequences to occur on the three buses” (2-3).

A second, related movement rejoins human elements to code that is otherwise consigned for use solely by the machines. As Brooks puts it, “a computer program is a message from a man to a machine. The rigidly marshaled syntax and the scrupulous definitions all exist to make intention clear to the dumb engine. . . . But a written program has another face, that which tells its story to the human user. For even the most private of programs, some such communication is necessary; memory will fail the author-user, and he will require refreshing on the details of his handiwork. . . . For the program product, the other face to the user is fully as important as the face to the machine” (164). Software products are rarely written once and for all time, never to be revised. Therefore, the message from human to machine is really often a long series of messages that can extend for decades throughout multiple discourse networks.

Running parallel to these musings about the materiality of code are judgments about the effects of using computers on thinking in its broadest sense. Horkheimer and Adorno alluded to ticket thinking, and IBM Hollerith punch cards epitomize this reduction of contextual, individual characteristics to discreet, digital designations that can be mechanically manipulated en masse. Black's description of the novel the use of Hollerith punch card computing by the Allies to assess the impact of bombing German civilian targets foreshadows a type of computer-aided thinking we take for granted today as integral to all institutional cognition. “The Bad Nauheim site was completely dependent upon Hollerith machines and Dehomag operators for its numerous calculations of bomb destruction and predictions of the resulting social disruption. The so-called Morale Division, staffed with a platoon of social scientists, psychologists, and economists relied upon the machines to quantify public reaction to severe bombing. Regular debriefing of civilians and experienced Gestapo agents regarding the dimensions of political dissension, as well as survey questionaires, were all reduced to researchable punch card data” (422).

Winner's arguments for why computing technology instills the absent mind and mythinformation was presented in the first chapter. Prior to him, Joseph Weizenbaum's *Computer Power and Human Reason* presented major philosophical issues that have been keeping ethicists busy ever since, and he is quoted frequently. Programmers sense the power of the computer by their ability to program it to do things, even if they do not know how it works. “The computer programmer's sense of power derives largely from his conviction that his instructions will be obeyed unconditionally and that, given his ability to build arbitrarily large program structures, there is no limit to at least the size of the problems he can solve” (105). Moreover, only partial understanding is needed to program, being an experimental, trial and error exercise, while propagating a myth of depth. “In effect, we all constantly use subroutines whose input-output behavior we believe we know, but whose details we need not and rarely do think about. To understand something sufficiently well to be able to program it for a computer does not mean to understand it to its ultimate depth” (108).

Weizenbaum is well known for testing this theory about perceived depth by writing the program *ELIZA* that simulated a Rogerian psychotherapist. His dismay at the positive response to this simplistic ruse by even trained professionals inspired him to write the book. The net result of these prejudices resembles and extends the Platonic critique of writing in *Phaedrus*. I presented his compulsive programmer in the introduction. “I have already said that the compulsive programmer, or hacker as he calls himself, is usually a superb technician. It seems therefore that he is not without skill as the definition would have it. But the definition fits in the deeper sense that the hacker is without definite purpose: he cannot set before himself a clearly defined long-term goal and a plan for achieving it, for he has only technique, not knowledge. . . . But since there is no general theory of the whole system, the system itself can be only a more or less chaotic aggregate of subsystems whose influence one one another's behavior is discoverable only piecemeal and by experiment” (118). The addition of autonomous behavior via conditional branching complicates the Platonic condition, for the computational world offers endless, chaotic aggregates of subsystems in which the writer possessing technique without knowledge advances through piecemeal experimentation—a fault, apparently, of both hard mastery and bricolage programming styles.

Logic is only a small component of ordinary human thinking, extending beyond the monolithic CPU paradigm by intuition and emotions into physical embodiment, a view supported by brain hemisphere studies (214). “The lesson here is rather that the part of the human mind which communicates to us in rational and scientific terms is itself an instrument that disturbs what it observes, particularly its voiceless partner, the unconscious, between which and our conscious selves it mediates. . . . We are capable of listening with the third ear, of sensing living truth that is truth beyond any standards of provability. It is *that* kind of understanding, and the kind of intelligence that is derived from it, which I claim is beyond the abilities of computers to simulate" (222). Invoking Whitehead's fallacy of misplaced concreteness, Weizenbaum explains that "we come to believe that these theoretical terms are ultimately interpretable as observations, that in the visible future we will have ingenious instruments capable of measuring the objects to which these terms refer" (222-223). This is why he shifts to an ethical stance against giving computers tasks demanding wisdom, for the combination of being able to write a program about something without deeply understanding it and misplaced concreteness has led to the perception that thinking machines are on the horizon.

The misattribution that programmers understand every detail of the processes embodied by their programs was realized by Wiener. "What Norbert Wiener described as a possibility has long since become reality. The reasons for this appear to be almost impossible for the layman to understand or accept. His misconcpetion of what computers are, of what they do, and of how they do what they do is attributable in part to the pervasiveness of the mechanistic metaphor and the depth to which it has penetrated the unconscious of our entire culture" (233). Thus lay persons believe programmers know every detail and theoretical basis, Weizenbaum contends, although their knowledge is much more sparse and brittle. In this sense, the fetishization of computers has become the concrete form of Horkheimer's eclipse of reason (252). Weizenbaum's response is that "the alternative to the kind of rationality that sees the solution to world problems in psychotechnology is not mindlessness. It is reason restored to human dignity, to authenticity, to self-esteem, and to individual autonomy. . . . But instrumental reason, triumphant technique, and unbridled science *are* addictive" (255-257).

The ethical recommendations with which Weizenbaum concludes *Computer Power and Human Reason* are surprising, if only because they have been largely ignored. First, treating everything as objects puts our souls at peril, he warns. “Is not the overriding obligation on men, including men of science, to exempt life itself from the madness of treating everything as an object, a sufficient reason, and one that does not even have to be spoken?” (260) Instead, “what we have gained is a new conformism, which permits us to say anything that can be said in the functional languages of instrumental reason, but forbids us to allude to what Ionesco called the living truth” (261). Fallacies of misplaced concreteness have made the functional languages of instrumental reason the only ones that can be legitimately used. Second, he makes a curious plea regarding the kinds of projects computer scientists ought to avoid. “There is, in my view, no project in computer science as such that is morally repugnant and that I would advise students or colleagues to avoid. There are, however, two kinds of computer applications that either ought not be undertaken at all, or, if they are contemplated, should be approached with utmost caution” (268). Obscene projects such as coupling an animal's visual system and brain are one category, a second being “projects that propose to substitute a computer system for a human function that involves interpersonal respect, understanding, and love . . . human functions for which computers *ought* not to be substituted” (269).

The second type of application to avoid or only undertake with careful forethought is “that which can easily be seen to have irreversible and not entirely foreseeable side effects,” especially if “such an application cannot be shown to meet a pressing human need that cannot readily be met in any other way” (270). In the years that have elapsed since the book was published in 1976, I am certain all of these morally questionable projects have been undertaken not just once but many times over, from violent video games and pornography to animal experiments and substitutes for human functions priming the robotic moment.[71] Civil courage in small contexts of governmentality, especially teachers of computer science, is the appropriate response for concerned philosophers of computing technologies. Again echoing Plato's critique of writing, Weizenbaum cautions that “almost anyone with a reasonably orderly mind can become a fairly good programmer with just a little instruction and practice. . . . Immature students are therefore easily misled into believing that they have truly mastered a craft of immense power and of great importance when, in fact, they have learned only its rudiments and nothing substantive at all. . . . When such students have completed their studies, they are rather like people who have somehow become eloquent in some foreign language, but who, when they attempt to write something in that language, find they have literally nothing of their own to say” (278). Finally, teachers of computer science need to do more than merely train, and they must resist their own “enormous temptation to be arrogant because his knowledge is somehow harder than that of his humanist colleagues” (279).

Besides being a philosopher of computing, on account of explicitly addressing the subject, I am calling Weizenbaum a *philosophical programmer* because his human-destined philosophical productions, i.e., the passages from *Computer Power and Human Reason* just discussed, were written in part as reflections on his own experience designing, writing, and testing software—*working code* as I will call it henceforth, appealing to the multiple

71 Weizenbaum gives the interesting choice of speech recognition as an application to avoid, contrary to Licklider; either it is too expensive or will lead to a surveillance state.

connotations of this phrase.[72] Moreover, he nearly qualifies as a *programming philosopher* because he designed, created, and exercised a substantial amount of machine-destined philosophical production, i.e., the *ELIZA* program to mimic a Rogerian psychotherapist, and used it extensively to study the human response to such a program.[73]

As a recent counterpart to Weizenbaum's work, I next want to consider David Golumbia's 2009 *Cultural Logic of Computation*. His first point is that computationalism, treating the mind itself as a computer, surprisingly underwrites traditional conceptions of humanity, society, and politics. Thus he rejects the notion of a historical rupture associated with the rise of electronic computing machinery; "networks, distributed communication, personal involvement in politics, and the geographically widespread sharing of information about the self and communities have been characteristic of human societies in every time and every place" (3).

Where Weizenbaum homes in on fallacies regarding computing power and the depth—or absence thereof—of program subroutines and engineers' understanding of what they are doing, Golumbia attacks the deliberate metaphor of programming languages. Although accepting definitions of computation as mathematical calculation that can stand for nonmathematical propositions, invoking Spivak, Landow and Derrida, he rejects the corollary assumption that languages are therefore reducible to codes, for the former rarely have a single, correct interpretation. "Programming languages, as Derrida knew, are codes: they have one and only one correct interpretation (or, at the absolute limit, a determinate number of discrete interpretations).

72 By multiple senses I mean Weizenbaum engaged in *working* as a verb, code as its object; second, working as an adjective meaning he wrote code that *works*, to be distinguished later from pseudo-code and code-work.

73 It is ironic that Weizenbaum's own *DOCTOR* project meets both of his criteria for an unethical application.

Human language practice almost never has a single correct interpretation. Languages are not codes; programming languages like Basic and FORTRAN and scripting languages like HTML and JavaScript do not serve the functions that human languages do. The use of the term *language* to describe them is a deliberate metaphor, one that is meant to help us interact with machines, but we must not let ourselves lose sight of its metaphorical status, and yet this forgetting has been at stake from the first application of the name" (19). Thus we are philosophically misled by a deliberate, utilitarian metaphor whose artificiality has been forgotten.

A second place where philosophical thinking about computing has been shunted involves attention given to Deleuze and Guattari's reflections on the virtual in the cultural study of computers, obscuring "their much more sustained and meaningful ideas that center on the term striation, and that clearly have computation as an historical abstraction, and not just material computers, as their object of analysis" (23). We do not want to admit overwhelming forces of striation in hegemonies of governmentality afforded by computationalism. Liberal political analysis instead favors two positions of democratizing technological determinism, or resisting through protocol. However, Golumbia drives home the point that computing is our fundamental governmentality, not just an industry and communications medium, raising crucial questions about "whether the shape, function, and ubiquity of the computing network is something that should be brought under democratic control in a way that it is not today" (25). Echoing Winner's mythinformation hypothesis, there is not enough evidence that computers bring the democratic actions liberal discourse proclaims, beyond social media effects, while there is plenty of evidence suggesting they bring increased authoritarianism, especially through surveillance, and corporate facism. Alluding to a theorist who will be introduced in a few pages: "Thus to [Alexander]

Galloway's dictum I offer this simple emendation: resistance through protocol, *and* against it” (26).

To buttress this view that computers foster authoritarianism, Golumbia develops an ironic critique of Noam Chomsky for lending philosophical weight to the development of artificial intelligence, despite Chomsky's long standing, subsequent involvement in liberal causes. A combination of Chomsky's charisma and the Cold War motivations to protect American cultural institutions gave computationalism its holding power, rather than belief in technological progress. “So some of the world's leading intellects have been attracted to it, precisely because of its cultural power; and so many of them have defected from it (just as so many have defected from Chomskyan generativism) because there exists a structure of belief at its core, one implicated not in technological progress but in the cultural structures of subjectivity” (53-54). Thus the Universal Translator from *Star Trek* reveals cultural beliefs about languages and future hopes of computer abilities grounded in them. “Through the discovery of some kind of formal principles underlying any linguistic practice (not at all just human linguistic practice), the Universal Translator can instantly analyze an entire language through just a few sample sentences (sometimes as little as a single brief conversation) and instantly produce flawless equivalents across what appear to be highly divergent languages” (85).

Golumbia combines his philosophical study with a rendering of the history of computing that suggests the linguistic theories of early computer engineers stem from their experience with computers rather than any extended study of linguistics. “But the early computer engineers—Turing, Shannon, Warren Weaver, even the more skeptical von Neumann—had virtually no educational background in language or linguistics, and their work shows no signs of engaging at all with linguistic work of their day. Instead, their ideas stem from their observations about the computer, in a pattern that continues to the present day" (86). The illegitimate analogy between code and language can be traced back to Weaver's famous 1949 memorandum later published as "Translation." "Weaver's principal intuition, that translation is an operation similar to decoding, was almost immediately dismissed even by the advocates of MT [Machine Translation] as a research program. . . . yet at some level, the intuition that language is code-like underwrites not just MT but its successor (and, strangely, more ambitious) programs of CL [Computational Linguistics] and NLP [Natural Language Processing]" (92-93). At its extreme, Weaver proposed erecting a "Tower of Anti-Babel" based on ideas about linguistics that he developed himself, "as if these fields had simply appeared *ex nihilo* with the development of computers" (87). No lesser a figure than the grandfather of digital humanities Roberto Busa picked up this same theme.

Yet the history of NLP and AI research is also full of defectors from the Chomskyan ideology. Golumbia notes that Terry Winograd abandoned SHRDLU, realizing it is a closed formal system requiring programming to extend meaning, "under the influence of Heideggerian thought such as that found in Dreyfus's writings, began to think about the ways in which computers as machines interact with humans and function in the human world, and to see language as a practice that can only be understood in context" (103). All the same, and more relevant to my studies, he regards the OHCO thesis, that texts are ordered hierarchies of complex objects, at least in its early days, as blindly reflecting the computational bias: "a gut feeling or intuition that computation as a process must be at the bottom of human and sometimes cultural affairs, *prior to the discovery of compelling evidence that such a thesis might be correct*" (106).

That digital humanists have swallowed XML as the preferred computer language for representing texts via the TEI merely transfers skepticism about the OHCO thesis to a more deeply rooted complicity. “It has been quite astonishing to see the speed with which XML and its associated technologies have not merely spread throughout humanities computing, but have become a kind of conversion gospel that serves not merely as motivation but as outright goal for many projects. . . . Humanities researchers don't produce databases: they produce language and texts” (116).

Golumbia's critique of the close association between established computing technologies and the empire stage of global capitalism presented by Hardt and Negri notes the differential benefits of IT for individuals, favoring the wealthy and powerful, and topics like surveillance and intellectual property. The balance sheet as a staple of business thinking before electronics is now extended to general workforce as spreadsheets. Their dehumanizing perception of time tracking and project management views of workers is insidiously tied into a culture of 'cool' associated with computer technologies.[74] Moreover, we need to be reminded that there were rhizomatic technologies before computerization, such as telephone networks, and the degree of control concretized in the computational infrastructure overwhelms the smaller, compensatory freedoms those networks afford. Indeed, perhaps more control by further regulating spaces that had not been formerly under the purview of surveillance. “What computers add to the telephonic connection, often enough overwriting exactly telephonic technology, is striation and control: the reestablishment of hierarchy in spaces that had so far not been subject to detailed, striated, precise control. . . . contrary to received opinion, it is the nature of our computers to territorialize,

74 See Jeff Rice's *Rhetoric of Cool* for an explanation of how this rhetoric operates.

to striate, and to make available for State control and processing that which had previously escaped notice or direct control” (153-154). What is more, the striation resulting from an expectancy of availability via every connected cellular phone interferes with the cultural importance of smooth spaces and times, for example evenings and weekends.[75]

Golumbia presents a philosophy of computationalism that equiocates reason with syntax, draws a correlation between rationalism and conservatism among philosophers, and explains how structural identification between programmers and the power elite sets up a majoritarian, white male capitalist image thriving on mastery.[76] Golumbia calls out his method identifying and describing a set of ideological pheonomena: “the functions of the discourse of computationalism in contemporary social formations; the imbrication of that discourse in a politics that seems strikingly at odds with a discourse of liberation that has more and more come to characterize talk about computers; and the disconnect between the capabilities of physical computers and the ideology of computationalism that underwrites much of our own contemporary investment in being digital” (221). Computers may have *savior* of many things that can be learned propositionally, but not *connaissance* of having that knowledge; and many things cannot be learned through propositional communication, especially embodied behaviors like midwifery. “It is not necessary to the becoming-self that a child use a computer, watch television, or even read books; it is necessary that she or he use language” (207).

75 At the same time, the enclosure of the workday is interrupted by local, personal communications devices outside the corporate infrastructure as well as those riding upon them, for example workstation Internet access.

76 Bill Gates and Steve Ballmer are criticized as quintessential techno-egotist male exploiters instantiating the Leviathan principle.

Mastery over computers is therefore seen as a poor compromise for those lacking social skills. “The concern Turkle and Weizenbaum (and even as sophisticated a writer as Galloway) do not seem to focus on what happens when the computer is the right instrument for the particular child when the computer itself, or even more profoundly, the social metaphor the computer offers, are ready-to-hand for the child set to adopt them” (207). Ready-to-hand technologies replicate their internal mania for classification in human beings. Where Kraft saw structured programming distorted to serve the needs of managers in their effort to de-skill the programmer workforce, Golumbia sees its successor, object-oriented programming, fitting within computationalism by emphasizing hierarchy, speciation, and categories.[77]

Returning to the theme of ideological phenomena, Golumbia argues we are overwhelmed by so many *presentist* views focusing on the latest tools. It “presumes that people in their social lives prior to the Internet did not exist as a broad network of autonomous social actors,” failing to see the prevalence of information throughout history, leading to belief in historical rupture and revolution, although philosophical classics like Hobbes' *Leviathan* foreshadow computationalism (213). Reaching a conclusion echoing Bauerlein, environments full of clearly indicated goals limit play and development of self regulation, which affects participation in democratic society. “Only by detailing exactly the nature of these forces do we hold out any hope of managing them” (155). His classic critical position leaves open whether *direct* participation is also a legitimate role of the scholar, whether in the form of managing or engineering, assuming the work of “detailing exactly the nature of these forces” can be accomplished separately. The anti-presentist

77 He lists a number of less-objectifying ontologies computer scientists have proposed, such as Aspect-Oriented Programming and Subject-Oriented Programming, but notes they have gotten little practical traction (210-211).

strain of Golumbia's thought seems to foreclose working through the philosophy of computing by doing things with new media, preferring traditional scholarly methods such as the thoroughly researched book under review. “What right have we to start new media when we have so poorly figured out what to do about the old ones?” (203-204). He appears to recommend that we recommence *old* media studies before continuing to theorize or produce new media, a tension Hayles, Bolter, O'Gorman and others identify as traditional humanities clashing with digital humanities. Consequently, Golumbia's powerful argumentation threatens to reduce an entire genre of liberatory manifestos into what I called foss hopes.[78]

What Golumbia does for computationalism, Alexander Galloway does for protocol, except he yanks it in the opposite direction, taking anti-presentist criticism in stride and accepting the isomorphism of the technological and socio-political, forcing a material and technically savvy understanding of technology. In other words, he takes Callon seriously. “Throughout the discussions on power, control, and decentralization, *Protocol* consistently makes a case for a material understanding of technology. . . . In short, the technical specs matter, ontologically and politically. . . . Code is a set of procedures, actions, and practices, designed in particular ways to achieve particular ends in particular contexts. Code = praxis” (xii). Most literature on protocol examines its relation to bureaucracy and law, such as Lessig's. But code is always process based. “The first point is that networks are not metaphors. . . . Networks are not tropes for notions of interconnection. They are material technologies, sites of variable practices,

78 Another set of theorists I am setting aside are proponents of democratization and foss including Feenberg, Stallman, Raymond, Torvalds, Himanen, and Harper.

actions, and movements. . . . A code, in the sense that *Protocol* defines it, is process-based: It is parsed, compiled, procedural or object-oriented, and defined by ontology standards" (xiii).

To study protocol and process-based code we are obliged to answer the question how does the Internet work with an analysis of both the technical and political dynamics of TCP/IP and DNS. "It asks how a particular type of network functions—the information networks that undergird the Internet. It shows how a network is not simply a free-for-all of information out there, nor is it a dystopia of databanks owned by corporations. It is a set of technical procedures for defining, managing, modulating, and distributing information throughout a flexible yet robust delivery infrastructure" (xv). A distributed network—"sets of nodes and edges, dots and lines"(xix)—is the native landscape of protocol, whose content is always another protocol; in Deleuzean terms, it is "an important diagram for our current social formation" (10). Protocological operations answer the question, "If we are indeed living in a post-industrial, postmodern, postdemocratic society, how does one account for political agency in situations in which agency appears to be either caught in networks of power or distributed across multiple agencies?" (xix) Its political character is displayed in moments of disconnectivity as other theorists emphasize glitches, blips and breakdowns; "the mere technical details, such as RFCs, suddenly become the grounds for contesting the way in which control takes shape in the materiality of networks" (xvi).

Significantly, the development of network protocols in the second half of the twentieth century leveraged *ontology standards* as the foundation of portability and layering. A term appropriated from philosophy used to describe the being of Being is now commonly used in computer science for agreed-upon code conventions that may not map onto any existing phenomena of the physical world. “Newer, more flexible markup languages such as XML have made it possible for researchers (be they biologists or engineers) to come up with a coding schema tailored to their discipline. . . . If layering is dependent upon portability, then portability is in turn enabled by the existence of ontology standards” (xxi-xxii). Layering and portability are crucial to TCP/IP networking.

Our prior familiarity with histories of computing and networking prepare us for the abridged narrative Galloway provides. He focuses on the distributed agency of Baran's packet-switching network situated in the surrounding equipment and the development of TCP/IP in the 1980s. Network packets themselves do not find their own ways through the network. “At the core of networked computing is the concept of protocol. A computer protocol is a set of recommendations and rules that outline specific technical standards. The protocols that govern much of the Internet are contained in what are called RFC (Request For Comments) documents” (5). Protocols form the core of networked computing governed by organizations like the IETF [Internet Engineering Task Force] and the W3C [World Wide Web Constortium], who freely publish them as RFCs and other document types.[79] “Etymologically it refers to a fly-leaf glued to the beginning of a document, but in familiar usage the word came to mean any introductory paper summarizing the key points of a diplomatic agreement or treaty. What was once a question of consideration and sense is now a question of logic and physics” (6). Questions of logic and

79 Both Abbate and Hafter and Lyon note that the ideal democratic spirit of the RFC method of proposing, debating, formalizing, and disseminating ontology standards and their specific instantiations in protocols was often distorted by the powerful personalities entrusted with managing them.

physics that, in contrast to the development of scientific knowledge, were often brought into being through compromise, fortuitous design, and usability testing as collaborators fit one piece after another into the collection of schemes emerging as the Internet.

A key theme of Galloway's book is the contrast between distributing TCP/IP and hierarchizing DNS (Domain Name System). The former, TCP/IP, exhibits an internal structure different from both centralized and decentralized networks, where a single node dominates or multiple nodes share control operations. “The network contains nothing but [quoting John Hall's *Internet Core Protocols*] intelligent end-point systems that are self-deterministic, allowing each end-point system to communicate with any host it chooses. Like the rhizome, each node in a distributed network may establish direct communication with another node, without having to appeal to a hierarchical intermediary. Yet in order to initiate communication, the two nodes must speak the same language” (11). The latter, DNS, is used to resolve domain names into numeric IP addresses to route network traffic around the planet. “All DNS information is controlled in a hierarchical, inverted-tree structure. Ironically, then, nearly all Web traffic must submit to a hierarchical structure (DNS) to gain access to the anarchic and radically horizontal structure of the Internet. . . . Because the DNS system is structured like an inverted tree, each branch of the tree holds absolute control over everything else” (8-9).

Galloway takes a philosophical stance by choosing to focus on bodies and the material stratum of computer technology rather than minds and epistemology. He follows the lead of Bazin, Barthes and Hayles analyzing material specific formal functions and dysfunctions of film, striptease, and digital media, rather than epistemologically minded theorists like Minsky, Dennett, Searle, and Dreyfus. “To steal a line from Foucault, it's not that protocol is bad but that protocol is *dangerous*. . . . I hope to show in this book that it is *through* protocol that one must guide one's efforts, not against it” (16). This is the quote Golumbia contests, siding with Weizenbaum that presentist obsessions with technology lure us into computationalist solutions to every problem. Surely Galloway understands this. In his broad periodization theory, protocol is the technical theory corresponding to Hardt and Negri's Empire as social theory. “At best, periodization theory is an analytical mindgame, yet one that breathes life into the structural analyses offered to explain certain tectonic shifts in the foundations of social and political life. My book implicitly participates in this game, mapping out certain details of the third, control society phase, specifically the diagram of the distributed network, the technology of the computer, and the management style of protocol” (26). Thus, there is danger in protocol like the danger of technology for Heidegger, and the danger of writing as *pharmakon* for Plato and Derrida. That is why efforts must be guided *through* protocol, because we are living under its sway as members of societies of control.

Galloway defines protocol as a universal description language of objects, the *chivalry* of objects. “Protocol is a language that regulates flow, directs netspace, codes relationships, and connects life-forms. . . . Protocol is always a second-order process; it governs the architecture of the architecture of objects. Protocol is how control exists after distribution achieves hegemony as a formal diagram. It is etiquette for autonomous agents” (74). Protocol is algorithmic, and may be centralized, distributed or decentralized. Centralized networks are hierarchical, operating from central hub; examples include the American military and judicial systems, and Foucault's panopticon. Decentralized networks contain distributed autonomous agents following system rules, exemplified conceptaully by Deleuze and Guattari's rhizome, and concretely by interstate highways, global airlines, and the Internet. Protocol is materially immanent, endogenous language that is indifferent to its content; “protocol does not follow a model of command and control that places the commanding agent outside of that which is being commanded” (51). This means more than the stored program concept, in which the same memory locations may contain instructions or data.

The immanence of protocol alludes to the fact that “since digital information is nothing but an undifferentiated soup of ones and zeros, data objects *are nothing* but the arbitrary drawing of boundaries that appear at the threshold of two articulated protocols” (51). Hafner and Lyon also provide an explanation comparing a manuscript flyleaf and packet header, revealing that this term, like so many others in electronics and computer technology, has an origin halfway between serious etymology and simulacrum. “They very word protocol found its way into the language of computer networking based on the need for collective agreement among network users. For a long time the word has been used for the etiquette of diplomacy and for certain diplomatic agreements. But in ancient Greek, *protokollon* meant the first leaf of a volume, a flyleaf attached to the top of a papyrus scroll that contained a synopsis of the manuscript, its authentication, and the date” (145-146). IP is responsible for routing datagrams and managing fragmentation, splitting datagrams into pieces and reassembling them. They are containers but they are themselves made of the same binary units as everything else.

The recommended way to envision how protocols operate in decentralized, distributed networks is via the Open Systems Interconnection (OSI) network layer model. In a footnote, Galloway explains that “the critical distinction is that the OSI model, my preferred heuristic, considers everything to be code and makes no allowances for special anthropomorphic uses of data. This makes it much easier to think about protocol. The other models privilege human-legible forms, whose reducibility to protocol is flimsy at best” (40). That is, the OSI model illustrates the schematism of perceptibility so crucial to Kittler's media theory, accepting no special, anthropomorphic uses of data, and affording an ontology of amalgamation of multiple processes occurring in multiple temporal orders of magnitude and systems exhibiting distributed control, fitting models described by Deleuze and Guattari (assembly, abstract machine, body without organs, lines of flight, strata) that I will develop as my culminating theoretical perspective, the procedural rhetoric of diachrony in synchrony. It is worth mentioning that the OSI model is often used to describe network phenomena, but in practice the Internet evolved around a simpler diagram of four basic layers. “The RFC on Requirements for Internet Hosts [RFC 791], an introductory document, defines the Internet as a series of interconnected networks, that is, a *network of networks*, that are interconnected via numerous interfacing computers called gateways” (38).[80] The Internet suite of protocols, according to this document, consists of the application layer (WWW), the transport layer (TCP), the Internet layer (IP), and the link (or media-access) layer (Ethernet).

A central concern of *Protocol* is how the decentralized, distributed control achieved in TCP/IP networks materially relates to Foucauldian biopower.[81] The materiality of protocols

80 The RFCs have yet to be thoroughly investigated as a discursive treasure trove for critical theorists.

81 Curiously, Galloway casts software as immaterial despite his argumentation aiming for the materiality of networks. “The niceties of hardware design are less important than the immaterial software existing within

related to biological life supports the idea that other protocols may be an affective, aesthetic force as well. Indeed, it is through the very concrete molecular structures of DNA that "*life*, hitherto considered an effuse, immaterial essence, has *become* matter, due to its increased imbrication with protocol forces" (82). He recounts Foucault's search for autochthonic transformation in the realm of words and things "that is immanent, particular, spontaneous, and *anonymous*" (83). Life merges with biopower and biopolitics, statistical knowledge about populations, species-level knowledge. However, his histories, and Foucault himself are "the rhetorical stand-in for the modern disciplinary societies, while Deleuze claims to speak about the future" (83).

The transformations of matter to media, life as code, immaterial soul replaced with aesthetized biometrics, are key to understanding rise of protocological control. Galloway steps through an analysis of vitalism in Marx's *Capital* to illustrate how material objects become aesthetic objects. It explains our sense in which capitalism is second nature, at once intuitive and naturalized, similar to Barthes' second order system of signification touched on at the beginning of this chapter. "The moments in Marx when he lapses into metaphor and imagery appear to be his own attempt at cinematography—that is, his attempt to aestheticize the vital forms contained in the body. . . . It is a 'layer' that has been folded back on itself such that it is simultaneously its core self and its own patina. It is both raw and coded" (92).

The intuitive capitalistic apparatus alluded to by this vitalistic imagery foreshadows protocol. "The use of vitalistic imagery, no matter how marginalized within the text, quite literally *aestheticizes capitalism*. It turns capitalism into media" (102). By materiality and

it. . . . Thus, the key to protocol's formal relations is in the realm of the immaterial software" (72).

aesthetics Galloway means a passage from non-media to media, and he declares that “protocol is a theory of the confluence of life and matter” (103). Taking a Deleuzean reading of Wiener, he argues that both people and machines are communicative organisms, which today live inside protocol. The essence of cybernetics is self-determinism of material systems, like Foucault's biopower. “This historical moment when life is defined no longer as essence, but as code is the moment when life *becomes a medium*. . . . One's lived experience was no longer tied to material realities, but instead was understood in terms of numbers a telephone number, a zip code, a social security number, an IP address, and so on. . . . It considers living human bodies not in their immaterial essences, or souls, or what have you, but in terms of quantifiable, recordable, enumerable, and encodable characteristics. It considers life as an aesthetic object” (111-113).[82] The move here is very important, for it transfers ontology from conceptual realms managed by philosophy to physical realms glimpsed by genetics, and to practical realms instantiated by computer engineering. The culminating theoretical framework I call procedural rhetoric of diachrony in synchrony applies this notion of life as an aesthetic object to the phenomenon cyberspace as collective intelligence, which in turn offers an actionable methodology called critical programming.

Protocol as controlling logic transcends institutions, governments, and corporations while being tied to them; membership in its technocratic ruling class is open. Jon Postel, one of the

82 To illustrate this transformation sensed by Foucault and Deleuze, Galloway offers a control matrix periodization diagram from feudal, modern, postmodern to future considering the orders of machine, energy mode, disciplinary mode, control diagram, virtue, active threat (resistance), passive threat (delinquency), political mode, stratagem, and personal crisis.

Internet's pioneers and principal RFC editor, credits its success to public documentation, free and cheap software, and vendor independence. Nonetheless, “because of the technical sophistication needed to participate, this loose consortium of decision makers tends to fall into a relatively homogeneous social class: highly educated, altruistic, liberal-minded science professionals from modernized societies around the globe” (122). Galloway documents how the loose affiliations of this technocratic ruling class were surprisingly localized: “of the twenty-five or so original protocol pioneers, three of them—Vint Cerf, Jon Postel, and Steve Crocker—all came from a single high school in Los Angeles's San Fernando Valley. Furthermore, during his long tenure as RFC editor, Postel was the single gatekeeper through whom all protocol RFCs passed before they could be published” (122). Moreover, much of the controlling software was written on and runs on Unix-based systems (123).

Consequently, beside the hype of the Internet ushering new degrees of freedom and voices to oppressed groups, it was constructed by a coalition that was connected to the American military-industrial-university complex. “Ironically, then, the Internet protocols that help engender a distributed system of organization are themselves underpinned by distributed, bureaucratic institutions—be they entities like ICANN or technologies like DNS. . . . Thus it is an oversight for theorists like Lawrence Lessig (despite his strengths) to suggest that the origin of Internet communication was one of total freedom and lack of control. . . . The founding principle of the Net is control, not freedom” (141). Galloway describes this mechanism as “anti-federalism through universalism . . . whereby universal techniques are levied in such a way to revert much decision making back to the local level” (140). In a long footnote he faults Lessig for not seeing control as endemic to all distributed networks governed by protocol, which must be partially reactionary to be politically progressive.

This affirmation of the reality of bureaucracy in the midst of rhizomatic networks represents a step beyond postmodernism appropriate to the post-postmodern network dividual cyborg I articulated in the second chapter. Galloway sums it up with an amusing reference to Barthes. “Perhaps I can term the institutional frameworks mentioned in this chapter a type of *tactical standardization*, in which certain short-term goals are necessary in order to realize one's longer-term goals. Standardization is the politically reactionary tactic that enables radical openness. . . . It is, as Barthes put it, our Operation Margarine” (143). In this short piece Barthes explains that “to instill into the Established Order the complacent portrayal of its drawbacks has nowadays become a paradoxical but incontrovertible means of exalting it. . . . A little 'confessed' evil saves one from acknowledging a lot of hidden evil” (40-41). Yes, standardization has its drawbacks, and we admit that our liberatory networking protocols were largely developed by a small clique of high school buddies in California. “What does it matter, *after all*, if margarine is just fat, when it goes further than butter, and costs less? What does it matter, *after all*, if Order is a little brutal or a little blind, when it allows us to live cheaply? Here we are, in our turn, rid of a prejudice which costs us dearly, too dearly, which costs us too much in scruples, in revolt, in fights and in solitude” (41). Linux Torvalds as inventor and self-proclaimed 'benevolent dictator' maintaining control over the Linux kernel tracks Galloway's comments about this necessary amount of control within a putatively free-wheeling emancipatory technology.

Control is now like a law of nature it is so imbricated in biopower and technological systems, forcing us to be thorough in our scientific approach to self reflection. Galloway labels the third part of the book *Protocol Futures*, beginning with a chapter titled *Hacking*. While hackers are now generally categorized as terrorists or anti-government libertarians, the truth is that “by knowing protocol better than anyone else, hackers push protocol into a state of hypertrophy, hoping to come out the other side. So in a sense, hackers are created by protocol, but in another, hackers are protocological actors par excellence” (158). Thus hackers practice what they preach in the very special sense that code is hyperlinguistic. “Code is a language, but a very special kind of language. *Code is the only language that is executed*. . . . So code is the first language that actually does what it says—it is a machine for converting meaning into action” (165). Because hackers operate in this hyperlinguistic realm, Galloway argues, they have a special connection to utopian visions. “Deciding (and often struggling for) what is possible is the first step in a utopian vision based in desire, based in what one *wants*” (168-169). This is a somewhat limited version of utopia, for it comes into being first and foremost in cyberspace. This is where the collective intelligence Pierre Lévy describes finds its fourth habituation, knowledge space, the cosmopedia of cyberspace. “Thus, I suggest that the hacker's unique connection to the realm of the possible, via protocol that structures itself on precisely that threshold of possibility, gives the hacker special insight into the nature of utopia—what he or she *wants* out of computers” (169).

We must be very clear here, to prevent this suggestion from being quickly debunked by the majoritarian response of the dumbest generation that just wants to play games and connect with their peers in social media. They are computer users, not hackers. The quintessential hacker

Galloway intends with this statement is Richard Stallman. For decades Stallman has been the philosophical voice of the free software movement, and an accomplished programmer and software architect of the GNU suite of operating system tools that, in conjunction with the Linux kernel, provides an alternative to commercial, closed source computing environments like Microsoft *Windows* and Apple's offerings. What Stallman wants out of computers and software is freedom, the freedom to use software for any purpose, the freedom to view and modify source code, and the freedom redistribute it. What Galloway, Lévy and Stallman intend, however, is not just freedom for humans to enjoy; *software itself wants to be free*. "Code does not reach its apotheosis *for people*, but exists within its own dimension of perfection. . . . Commercial ownership of software is the primary impediment hated by all hackers because it means that code is limited—limited by intellectual property laws, limited by the profit motive, limited by corporate 'lamers'" (170).[83]

What should be stressed, Galloway insists, is pro-protocolism above anti-commercialism. Protocol is open source by definition, for this term is "given to a technology that makes public the source code used in its creation. That is to say, protocol is nothing but an elaborate instruction list of how a given technology should work, from the inside out, from the top to the bottom, as exemplified in the RFCs" (171). Thus the hacker outcry of the Digital Millennium Copyright Act of 1998 is "one of control over a specific technical knowledge, not potential piracy of DVD

83 Stallman is also a prime example of an individual who should learn to accept a little margarine. His extreme views of using only GNU-compliant devices down to the platform level have marginalized him from scholarly communities and other technologists who see free, open source software as a crucial but not exclusive ethical position when it comes to which software applications and computer models they personally use.

media" (172). By prohibiting the pursuit of knowledge itself, the DMCA is diametrically opposed to how hacking reveals ways of using code creatively. "What hacking reveals, then, is not that systems are secure or insecure, or that data wants to be free or proprietary, but that with protocol comes the exciting new ability to leverage possibility and action through code" (172).

The remaining chapters of *Protocol* address what Galloway calls tactical media and Internet art, and could be presented in the following section on Software Studies. The comparison between protocol and empire implicates it in the management style of the ruling elite as well as the enemies of power, as Foucault notes of biopower generating new forms of control and delinquency. The prediction and explanatory aside that Galloway makes, "it is very likely if not inevitable that the core Internet protocols, today largely safe from commercial and state power, will be replaced by some type of proprietary system. (The fact that Microsoft has not yet replaced TCP/IP with a commercial product of its own is one of the miracles of computer history. Chances are this will happen very soon.)" (244) reflects the times in which the book was written, the early 2000s when Microsoft was still the big enemy representing the establishment and free software the liberator.

Comparing protocol to the market economy permits analogous critical questioning; however, the special features of code being executable and autonomous requires further analysis and differentiation. Like Golumbia, who criticizes his fellow digital humanists for their fixation on XML without suggesting any alternative, more philosophical programming languages, Galloway seems to stop on the very threshold at which they propose new forms of critical thinking may be enacted. The reason, I claim, is because they are not themselves practicing programmers. While Galloway asserts that his focus is on bodies and the material stratum of computer technology rather than minds and epistemology, he notes a few pages later that “I attempt to read the never-ending stream of computer code *as one reads any text* (the former having yet to achieve recognition as a natural language), decoding its structure of control as one would a film or novel” (20). Weinberg will be the first to remind us that programmers do not read code like a novel, from beginning to end. There is also a telltale footnote on page 171 revealing curiously ill-informed mischaracterizations of Linux as an operating system, and foss as freeware, reminiscent of a similar inelegance perpetrated by Deborah Johnson in a footnote of *Computer Ethics, Third Edition*: “Perhaps, the best example of successful shareware is the Linux operating system” (160).

I would be remiss in presenting those who I feel are significant philosophers of computing technologies if I did not include Hayles, whom I have cited extensively already. She more than any of the others combines detailed historical research and accurate exposition of scientific theories with insightful readings from classic literary works, science fiction, film and new electronic literature, in what she calls 'tutor texts' to help ease students of the liberal arts into the digital humanities, and engineering-minded folk into the liberal arts. Her contributions have propelled a number of scholars into the limelight of relatively new disciplines like software studies and code studies, to which I now turn.

Software Studies

David Berry, in *The Philosophy of Software,* identifies interesting new forms of scholarship around software studies, cultural analytics, and critical code studies, whereas Hayles makes more detailed distinctions in *How We Think*: platform studies, media archeology, software engines, soft authorship, genre analysis of software, graphical user interfaces, digital code literacy, temporality and code, sociology and political economy of the free software and open source movement. Berry argues understanding software avidities affects human freedom, so therefore it is worthwhile to study, imbricating classic Socratic questioning for tracing agentic paths constituting human experience. His aim is to understand how being-in-the-world is made possible through application of computational techniques manifested in processes touched by software and code (11). Most code experience is visual rather than haptic, and the forms of computational literacy that Manovich, Bogost, Gee, Fuller, and other theorists are promoting has also been hinted as a new academic goal, from the distinction between close versus distant reading of texts to claim that "computational approaches facilitate disciplinary hybridity that leads to a post-disciplinary university that can be deeply unsettling to traditional academic knowledge" (26).

Software studies emerged as an academic discipline as the focus of new media studies turned more and more to its computational aspects. Thus its agenda is markedly broader than software histories, yet its adherents deliver more nuanced philosophical investigations than those of the previous section for whom computing technologies are paramount but not the focal objects of their studies. The chronologically ordered *New Media Reader* edited by Noah Wardrip-Fruin and Nick Montfort documents key writings that constitute, according to Janet Murray's introduction, "the first comprehensive effort at establishing the genealogy of the computer as an expressive medium. . . . [I]t was not until the 1980s that practice became self-conscious enough to allow for a serious discourse about digital artifacts. . . . Newer media such as photography, radio, film, and television could now be seen in the longer history that stretched back beyond the printing press to oral composition and the invention of writing" (3-7).

Lev Manovich, in writing the second introduction to this volume, offers eight propositions about new media that foreshadow his own transition to godfather of software studies: its focus on the cultural and computing, the use of computer technology as a distribution platform, new media as digital data controlled by software, as the mix between existing cultural conventions and the conventions of software, and as faster execution of algorithms previously executed manually or through other technologies (16-22). Moreover, each of the five trends Manovich presents in *The Language of New Media* are also key characteristics of computing technology, and we can compare his use of three approaches to studying cinema to techniques to study the history of software, programming, and the software industry. But Manovich is clear on this point: "to understand the logic of new media, we need to turn to computer science. It is there that we may expect to find the new terms, categories, and operations that characterize media that become programmable. *From media studies, we move to something that can be called software studies—from media theory to software theory*" (48).

New media exhibit a cultural layer and a computer layer, such that "the computerization of culture gradually accomplishes similar transcoding in relation to all cultural categories and concepts. That is, cultural categories and concepts are substituted, on the level of meaning and/or language, by new ones that derive from the computer's ontology, epistemology, and pragmatics" (47). Thus the fractal structure of many new media objects that extends down to the character or pixel level results from encapsulation protocols and modularity moreso than from an aesthetic style. Applying the Whorf-Sapir hypothesis that the code of natural languages determines thinking to computer-mediated culture, for example, we see that a command line interface is closer to literacy than a point and click GUI. Nonetheless, to Manovich our subjective experience even of computer interfaces, especially games, is deeply influenced by cinematic conventions. “Dynamic, real-time, and interactive, a screen is still a screen. Interactivity, simulation, and telepresence: As was the case centuries ago, we are still looking at a flat, rectangular surface, existing in the space of our body and acting as a window into another space” (115).

The difference is that the computer fulfills the promise of cinema as visual Esperanto: mobile camera, framing, scrolling over the interface, joined with real time response makes virtual reality creation by digital compositing a qualitatively new visual media. Manovich argues this audio-visual-spatial culture is a distinct phenomenological realm, and at exactly the center of *The Language of New Media* he presents the notions of spatial and ontological montage “by establishing a logic that controls the changes and the correlation of values on these dimensions . . . [and] the coexistence of ontologically incompatible elements within the same time and space” (158). Finally, digital media offers stylistic montage by allowing filmmakers to hide or foreground stylistically diverse formats, such as in *Forest Gump*: “this simulation of different film and video artifacts is an important aspect of its narrative system. . . . the shots in his films that combine live-action footage with graphic elements position all elements on parallel planes; the elements move parallel to the screen” (159).

Manovich joins Turkle and others who give examples of how computer technology instantiates otherwise vaporous postmodern concepts. He argues that software like *Photoshop* made postmodernism possible for the masses by standardizing and making it easier to construct media objects from existing, commercially distributed elements. Other facets of software-based media transcend the traditional, cultural domain of representation. Telecommunications, teleaction, and telepresence do not generate familiar representational objects. “By foregrounding telecommunication, both real-time and asynchronous, as a fundamental cultural activity, the Internet asks us to reconsider the very paradigm of an aesthetic object. . . . If a user accessing information and a user telecommunicating with other(s) are as common in computer culture as a user interacting with a representation, can we expand our aesthetic theories to include these two new situations?” (163-164). The ability to teleport is a unique, new media feature previously requiring action on part of a human. Hyperlink signs also take on the ability to teleact, not just signify.

Granting “new media embeds cinema-style illusions within the larger framework of an interactive control surface,” (211) moreover, to Manovich computer interfaces are aggregate versus systematic spaces, and we should look at software tools, organization, default settings before finished objects the same way “art historians and literary film scholars have traditionally analyzed the structure of cultural objects as reflecting larger cultural patterns (for instance, Panofsky's reading of perspective)” (258). In this view, “we can understand place as a product of cultural producers, while non-places are created by users; in other words, non-place is an individual trajectory through a place” (280). The criticism that even putatively free wheeling, democratizing virtual places have been territorialized by commercial software and their backing interests has been popular among foss evangelists for a long time.

Going a step beyond Manovich, Ian Bogost approaches software studies from a background in literary criticism and his practical experience as a software developer. In *Unit Operations: An Approach to Videogame Criticism*, he positions himself as a premier philosopher of computing, providing an ambitious methodology useful to humanists and technologists by exploring relations of computation, literature, and philosophy via this “general conceptual frame for discrete, compressed elements of fungible meaning” (xiii). Key theoretical advances fueling his approach are “the introduction and adoption of *object technology* (OT) in software engineering, and the advent of complex adaptive systems theory in the natural, information, and computer sciences” (x), although he situates the origins of unit analysis in classical philosophers from Aristotle to Spinoza, and contemporary philosophers Badiou and Harman. By drawing connections between the compression of representation evident in structuralism and poststructuralism with advances in computation, he is able to “use the four core principles of object technology to critique many of the popular academic works on digital media (Lev Manovich, George Landow, Jay Bolter) and genetics (Darwin, the Human Genome Project, Dawkins)” (xiv).

The second facet of Bogost's method he calls *procedural criticism*, which resembles Hayles' Media Specific Analysis by identifying “configurable expression in multiple media . . . showing how these texts function and interact through unit operations” (xiv). Following procedural criticism is procedural subjectivity, which explores “the interaction between embedded representation and subjectivity, arguing that meaning in unit-operational systems arises in a place of crisis between configurative representation and subjectivity” (xiv). This maneuver allows criticism to “vault videogames toward a status higher than entertainment alone, focusing specifically on an analysis of *Star Wars Galaxies* as a social text” (xiv). The fourth part of his method draws on relations between Deleuze and Guattari's schizoanalysis and complex network theory to provide a way to analyze freedom in large virtual spaces like the videogame *Grand Theft Auto 3*.

The notion of unit operations exercises some themes already covered with respect to Derrida's *morceau*, taking bite-sized pieces for analysis, although Bogost looks to Graham Harman's object-oriented philosophy as the basis for extending Heideggerian humanist philosophy into machine being. It “interprets Heidegger's analysis of *Zuhandenheit*, or readiness-to-hand, as a quality available to entities other than *Dasein*. . . . understanding units as objects is useful because it underscores their status as discrete, material things in the world. . . . Harman insists on inanimate objects as necessary subjects for philosophy; while I include in my understanding of units ordinary objects such as the ones Harman favors (person, hammer, chandelier, insect, or otherwise), I also claim that units encompass the material manifestations of complex, abstract, or conceptual structures such as jealousy, racial tension, and political advocacy” (5). Bogost prefers units to objects to avoid connotations stemming from computer science and to include material manifestations of complex structures.[84]

Unit level analysis prepares us for dealing with the chaotic behavior of complex, distributed systems such as TCP/IP networks, whereas “unlike complex networks, which thrive between order and chaos, systems seek to explain all things via an unalienable order. . . . Stability, linearity, universalism, and permanence characterize system operations” (6). The stable experience inside a single process von Neumann computer permits holistic, systemic analysis, but the contemporary situation requires a different, wandering perspective, which Bogost eventually associates with the *flanuer*, as well as the style Heidegger deploys in “The Question

84 His recent *Alien Phenomenology* takes up this subject again, and Harman and Badiou are featured theorists.

Concerning Technology.” As Bogost explains, “we cannot escape systems, but we can explore them, or understand ourselves as implicated in their exploration. Heidegger's essay on technology is structured as a haptic analysis, akin to a walk in the woods, by which the stroller happens upon matters of interest. He takes this casual encounter as a paradigm for resistance. Like Heidegger's logic of the promenade, unit operations meander, leaving opportunities open rather than closing them down” (7).

Systems and units share operations as “a basic process that takes one or more inputs and performs a transformation on it,” but “in the language of Heidegger, unit operations are creative, whereas system operations are static. In the language of software engineering, unit operations are procedural, whereas system operations are structured” (7-8). The difference is subtle but the passage through Kraft's sociology of programming management helps. All computers based on Burks, Goldstine, and von Neumann's stored program architecture rely on sequential processing, and from that basic model extraordinarily complex systems were designed. Structured programming attempted to formalize the commonly employed algorithms to make programming work resemble a factory model, with a very small number of lead programmers at the top. Systems operations epitomize this hierarchical organization of labor, whereas unit operations retain interest in the processes employed therein. “The movement away from systems thinking is really a movement away from the simple, orderly, static categorization of things” (8).

Besides Heidegger, Spinoza is the philosophical antecedent preparing the way for these more complex categorizations. “Spinoza's philosophy sets up a network-like superstructure for almost any kind of material relation. . . . The crucial seed that Spinoza plants is that of innumerably re-creatable relations between objects. Such language looks forward to forms of material relation like Valera and Maturana's autopoiesis, as well as the dynamic structure of software information systems” (9). As Harman provides the contemporary treatment of Heidegger, Alain Badiou's application of set theory to ontology updates Spinoza. “Badiou's philosophy offers a concept of multiplicity that simultaneously articulates coherent concepts and yet maintains the unitarity of their constituents. . . . This concept of membership, borrowed from set theory, forms the basis of Badiou's ontology: 'To exist is to be an element *of*'” (11). Badiou's concept of the *count as one* gives existence via set membership, whatever it may be. “As a process or a frame for *a* multiplicity, the count as one *produces a particular set*; it takes a multiplicity and treats it as a completed whole” (11). Something as simple as the return value of a function call instantiates Badiou's count as one in software by programming.

In a move made by a number of other theorists, Bogost leverages everyday discourse from computer science to instantiate philosophical concepts that have been very difficult to articulate in natural environments. By adding the computer science notion of procedurality, “the computer's special efficiency for formalizing the configuration and behavior of various representative elements,” to form units, we achieve “a ligature between computational and traditional representation, creating a common groundwork for understanding texts of all kinds as configurable. The count as one is the closest extant philosophical concept to what I am calling unit operations: an understanding, largely arbitrary, certainly contingent, of a particular situation, compacted and taken as a whole” (13). The count as one is the metaphysical or, in the case of computing, *real* operation by which unitary wholes arise. Taken together, Bogost defines unit analysis as “the general practice of criticism through the discovery and exposition of unit operations at work in one or many source texts. . . . Each medium carries particular expressive potential, but unit analysis can help the critic uncover the discrete meaning-making in texts of all kinds” (15).

The first extended example he gives analyzes the movie *The Terminal* to reveal “itself not as a film about a man struggling against governments for his identity, but as one about various modes of *waiting*” (16). That is, he finds a common theme of different modes of waiting exhibited by various characters interacting along different time horizons represented in the film, opening considerations of the 'uncorroborated wait' emblematic of bureaucratic and political waiting by the protagonist. He claims the novelty of this critical approach “privileges discrete components of meaning over global narrative progression,” (19) making it ideal for open-ended phenomena like computer games and electronic literature that defy the traditional narrative package of the novel or film. It is only a short step further, or back into its roots, that unit analysis be applied to software studies. The difference between ontologies in computer science and philosophy, Bogost notes, is that the former enable functional relationships between constituent parts of software systems, but do not specify them. “Unit operations, however, strive to articulate both the members of a particular situation *and* the specific functional relationships between them” (14). Examining source code in the manner Weinberg suggests, and the studying the behavior of running software, which Bogost prefers, closes this hermeneutic circle.

Applying unit analysis to software studies begins by acknowledging that the von Neumann architecture initiated a unit versus system operational model of computing by affording discrete divisions of control, memory, input, and output. Conditional control transfer universalized various operations that became standard subroutines and functions from which larger programs are built. “Stored-programming makes units of each program reusable and executable based on programmatic need rather than physical arrangement" (25). Thus, although we speak of computer systems today, their underlying model is reliant on the conjunction of sundry unit operations, exemplified by the Unix operating system model of a kernel surrounded by utility programs supporting user applications. Likewise, the ontological implications of both structured and object-oriented programming (OOP) encourage a unit analysis approach to designing software. Bogost cautions that "the promise of universal computation is precisely what the logic of unit operations must take care to exceed. If we have indeed begun to represent and understand the world via discrete, encapsulated logics that both include and exclude a variety of conditions, then we must understand how these unit operations work, where they attach to one another and to our understanding of the world, and how we should approach them as users, creators, and critics" (30).

Component objects are developed to address the growing mass of software libraries to encapsulate intellectual capital in black boxes. Structured programming promoted the abstraction of common functions into libraries of precompiled code that could be encapsulated, sold, and distributed as commodity units. Object technology (OT) took this idea a step further to encapsulate software data structures and their related functions in a similar manner. "OT attempts to close the gap between human experience, its programmatic representation, and its computational execution. Computational systems thus strive to create more successful implementations of automated human needs" (38). The formal definition of OOP includes four properties: abstraction, encapsulation, polymorphism, and inheritance.[85] Heeding Manovich's

85 The engagement with Stroustrup and other computer scientists credited with developing OOP is reserved for future work.

observation that "transcoding is new media's tendency to computerize aspects of nondigital organization, conflating their structures with the structure of the computer itself," Bogost suggests that "unit operations can help us expose and interrogate the ways we engage the world in general, not just the ways that computational systems structure or limit that experience" (39-40).

Discreteness is a principle property of unit operations relating to encapsulation. He provides an extended example of unit operations in the banking customer-account relationship and licensing of literary and cultural artifacts. "By encapsulating the customer-account relationship, and by reinforcing the behavior of banking through software systems, the banking industry has succeeded in remapping the material reality of capital exchange with its objectified, encapsulated object equivalent. In other words, our relationship with banks have become unit operations" (41). That is, the scripted ways that humans now engage with institutional capital have firmed up from the formerly flexible interactions at the teller line and drive-up window, to well defined unitary transactions exemplified by the ATM and now online banking applications. This formation of what Kitchin and Dodge call "code/space", regulating both the physical world and virtual places, has become a primary focus of software studies. Cultural space itself has succumbed to unit operations that are both helped and distorted by the computer technologies surrounding them. "Licensing is an example of the fungible use of a unit operation in the cultural, commercial, and legal registers" (42).[86] We encounter the same form of long, unreadable

86 Tanaka-Ishii brings up the distinction between family relations and licenses often used in educational settings to distinguish inheritance and interface in OOP, thus leveraging the unit operations each term implies: "A frequent classroom metaphor used in teaching programming languages is that classes connected by `extends` create a family system, whereas abstract data types defined by `implements` create a license system" (81).

license agreements that we must acknowledge through the painless click-through operations with software, music, literature, videos, games, and so on.[87]

Bogost does not analyze all four fundamental concepts of OT, giving very brief but believable starting points for abstraction and polymorphism, where “one can see a correlation between abstraction and the transcendental signified, or between polymorphism and intertextuality”, and ignoring inheritance altogether, declaring “it is not my wish to show how culture, philosophy, and critical theory can be made to 'look like' object technology” (45-46). He returns to objects return in *Alien Phenomenology*, but far distanced from the computer science conceptualization, and more indebted to contemporary ontologists like Harman.[88]Bogost prefers to consider the procedural over object aspect by gesturing to Murray's notion of procedural authority, so that “critics and creators can use a common toolkit to approach art and cultural objects that have equal home in both the worlds of the literary and the technological; we can understand unit-operational systems both inside and outside technology” (46). Indeed, the remainder of *Unit Operations* foregrounds procedurality in videogames, and *Persuasive Games*, published the following year, recycles much of its content.

---

87 Playing along the lines of cultural unit operations, Bogost points out how Landow's reading of Derrida's “Signature Event Context” privileges theory over technology, but well exemplifies the fungibility of Derrida between philosophical and technological discourse, as if the quoted passage could pass off as von Neumann or Kay (42).

88 This thought of not walking away from the other fundamental concepts of OOP relates to the exam question invoking Tanaka-Ishii considering limitations of representation for computational systems by semiotic analysis of computer programs as systems of signs exemplifying how theories from other disciplines can frame and shape our understanding in other domains, which is what Hayles finds remarkable in Malabou's study.

The procedural focus means discerning the unit operations at work in a cultural production, whether it is a literary work or a videogame. The materiality of game engines (the underlying, shared code used by a number of commercially discrete products) lends videogames to unit analysis moreso than other literary objects. Besides structuring possible narratives, they create similarities between works based on literal sharing of components, exemplified by a number of similar Atari VCS games. “Taken as a unit of gameplay, *Tank* took the notion of vector geometry as a mediator of competition between two players. . . . *Tank*, *Pong*, and *Combat*'s relation to one another is far stronger than interpretive notions like intertextuality or new media concepts like remediation allow” (58). More sophisticated games today also exhibit characteristic unit operations; “first-person shooter game engines construe entire gameplay behaviors, facilitating functional interactions divorced from individual games” (57). The activities in which players engage—selecting weapons, navigating their play field, strafing fire, crouching, and so on—have become well established conventions that are taken for granted.

This formal relationship between games due to shared core portions of code accentuates the merger of functionalism and materialism, and thus there are crucial differences between unit operations and traditional literary relations: they are legal, not discursive, and material, not psychoanalytic. “The discursive carriage of the FPS [first-person shooters] will change further as game engines, tools, and libraries move beyond killing, racing, and visual effects to emotional conflict, jealousy, and disappointment” (63). Reflecting on ideological contexts carried by legal and material relations is the most important aspect of videogame studies because they are where code operates upon the world via rhetorically engaging humans. “Videogames require critical interpretation to mediate our experience of the simulation, to ground it in a set of coherent and expressive values, responses, or understandings that constitute effects of the work. In this process, the unit operations of a simulation embody themselves in a player's understanding. This is the place where rules can be grasped, where instantiated code enters the material world via human players' faculty of reason” (99).

The crucial task is exploring game rules manifest in player experience. Other theorists besides Bogost have taken up this effort.[89] Clearly there is more to software studies than videogame criticism, however. In terms of its short history as a scholarly discipline, the software studies trajectory moves from Manovich's initial cinematographic focus to games and electronic literature, and finally—as if it were forgotten—to types of software applications, programming languages, algorithms, and data structures. *Software Studies: A Lexicon*, edited by Matthew Fuller, advertises its scope to include “algorithms; logical functions so fundamental that they may be imperceptible to most users; ways of thinking and doing that leak out of the domain of logic and into everyday life; the judgments of value and aesthetics that are built into computing; programming's own subcultures and its implicit or explicit politics; or the tightly formulated building blocks working to make, name, multiply, control, and interrelate reality” (1).

While an excellent source of introductory material to a variety of topics to illustrate the scope of the field and its proponents, it does not provide a sustained theoretical trajectory. Fuller's prior work, *Behind the Blip: Essays on the Culture of Software*, parallels Manovich and Bogost in defining software studies and laying out its tasks, which to him, writing in the early 2000s still at the height of the perceived Microsoft monopoly and when foss was beginning to garner mainstream attention, ought to be aimed at undermining software oligopolies. Moreover,

89 I refer to the work of Frasca, Murray, Turkle, Gee, and Wardrip-Fruin, among others.

he is reacting to the perception “that the vast majority of research and production in this area remains concerned with imposing functionalist models on all those systems that cohere as the user. . . . This is the fatal endpoint of the standard mode of HCI. It empowers users by modeling them, and in doing so effects their disappearance, their incorporation into its models” (13). Software criticism, therefore, ought first and foremost to pursue the question of “what would it mean to incorporate an explicitly wider notion of such processes into software – to reinfuse the social, the dynamic, the networks, the political, communality (perhaps even instead of, or as well as, privacy) – into the contained model of the individualized user that HCI has us marked down for?” (14).

Importantly, Fuller also identifies accounts by programmers for insights into understanding software as culture, for they are “a direct route to the cultural backbone of classical idealism” (15). I cast these perspectives as those of *philosophical programmers* to distinguish their less refined in terms of formal training in academic criticism but more authentic responses on account of being close to the machine, which he blithely puts as “access to and understanding of this beauty is allowed only to those souls that are themselves beautiful” (15).[90]

Fuller contends that most critical positions, when examining technologies, make their judgments on “the basis of a category or class of objects, rather than specific instances of that class,” blaming “pretensions to timelessness” for not wishing to tarry with the details of individual software components that are enmeshed in their particular milieu (16). Therefore, he makes the interesting suggestion of considering alternate timescales recommended by Donald

90 I refer to the collections of interviews and short pieces by Shasha and Lazare, Lammers, and especially Oram and Wilson's *Beautiful Code*.

Knuth, “when he proposes a deceptively simple task for computer scientists: Analyze every process that your computer executes in one second. . . . Why not contaminate this simple telling of the story of what goes on inside a computer with its all-too-cultural equivalent? The transcript of the contents of a mind over one day” (17). Both should be within the scope of software studies: technical analysis of machine operations at alien timescales, and everyday ethnography of programmers at work.

Another reason software studies have not taken off in the humanities may be an aversion to the electronic that is the hallmark of conceptuality. Fuller notes a systematic suspicion of the electronic throughout Deleuze's writings, despite the suitability of his and Guattari's work for theorizing software as a form of subjectivity. “The conceptual personae that Deleuze and Guattari so suggestively propose in *What Is Philosophy?* can be read as a proposal for an understanding of software as a form of digital subjectivity – that software constructs sensoriums, that each piece of software constructs ways of seeing, knowing, and doing in the world that at once contain a model of that part of the world it ostensibly pertains to and that also shape it every time it is used” (19). Their work helps us view computers as assemblages, and reject the notion that a particular level is definitive, and accept combinations with other systems as aspects of variable ontology—themes Bogost draws from his engagement with Harman. However, computers and software are cast in the same generic category as television and other electronic media. Yet Fuller discovers that when Deleuze and Guattari describe a 'thought synthesizer' in *A Thousand Plateaus*, they suggest “by assembling modules, source elements, and elements for treating concepts (oscillators, generators, and transformers), by arranging microintervals, the synthesizer makes conceptualisable the philosophical process, the production of that process itself, and puts us in contact with other elements of matter. In this machine composed by its materiality and force, thought travels, becomes mobile, synthesizes” (21-22).

Perhaps this aversion to detailed engagement with computers and programming languages reflects the inadequacy of the technological milieu to embody the ideas that are portrayed by this fantasy device. Fuller proposes that instead of looking for software criticism in the traditional areas, “we pay attention to some practices within software production that emerge with and through thought out of whack with its simple reproduction” (22). This is the working code of those I call *programming philosophers*—software engineering deliberately performed to investigate or highlight philosophical questions arising from technology itself, responses arising from the same domain in which the questions are raised. The three categories Fuller identifies are critical software, social software, and speculative software.

Critical software is deliberately designed to foreground default understandings of software “by using the evidence presented by normalized software to construct an arrangement of the objects, protocols, statements, dynamics, and sequences of interaction that allow its conditions of truth to become manifest” (23). “A Song for Occupations” maps out the entire Microsoft Word interface “to reveal the blue-grey labyrinth in which writing is so happily lost” (23). Richard Wright's *Hello World* CDROM makes “a comparative analysis of the interfaces and data structures—and consequent ways of knowing, seeing, and doing—of various video-editing and -effects packages such as Quantel, After Effects, and Flame” (23). The other type of critical software appears to run “just like a normal application, but has been fundamentally twisted to reveal the underlying construction of the user, the way the program treats data, and the transduction and coding processes of the interface” (23). This may be done by using a game engine to create behind-the-scenes views of the usual play, or manipulating the settings of commercial software to starkly change the user interface or grant access to normally hidden components. “What this work does is make apparent the processes of normalization operating at many scales within software—the ways in which, for instance, millions of separate writing acts are dedifferentiated by the various layers of a word processing program” (23).[91]

The second category, social software, has been created by and is used by those seeking to live outside the narrowly engineered subjectivity of mainstream software, developed through user interaction, especially in foss. Fuller immediately points out the internalist tendency of most free software, where “the relation between its users and its developers is so isomorphic that there is extreme difficulty in breaking out of that productive but constricted circle” (25). Using volunteer work and miniscule funds in its early years, it struggled to be merely productive, let alone “begin to set technico-aesthetic agendas which open and set flying the ways of sensing, knowing, and doing built into proprietary software” (25). Foss hopes becoming conceptually stalled, therefore, surround a promising, real component of software production; “there is a far more important need to recognize and find ways of coming into alliance with forms of intelligence that are excluded from the depleted culture of experts” (27).

Fuller considers a poetics of connection as one means by which experiments with social software can involve artists outside the culture of experts who work within the companies that regulate mobile phone networks. There has been “no substantially innovative work by artists using mobile phones alone—there is no way of messing with the architecture” (27). Yet

91 Bogost surveys similar programs in both *Unit Operations* and *Persuasive Games* in his very deliberate attempts at software criticism. It would be interesting to examine early hacks of simple software such as Apple // games, and software that reveals datastreams for the secret lives of devices.

teenagers have overrun mobile communications, and have developed their own “dimension of relationality rather than of territoriality. It is in their capacity to generate a poetics of this connection that they have reinvented this technology” (28). Mongrel's *Linker* is a software application for rapidly producing multimedia collages, and exemplifies the other sort of social software besides foss that Fuller hopes may be used to generate new forms of linking by not imposing a ready-made set of tools like Microsoft *Word* or Macromedia *Director* on the users.

Third, there is speculative software, which functions like the best fiction to “identify more precisely definable, albeit massively distributed and hierarchised, conflictual, imaginal, and collaborative relations” and as 'mutant epistemology' exploring “the potentiality of all possible programming” (29-30). Fuller introduces speculative software by quoting from Ellen Ullman's *Close to the Machine,* which he considers the “best account of the lived experience of programming” (29). Ullman contests the notion that computers are neutral, especially for programmers, who are always adding more code to their creations. At one point she exclaims “Finally, we arrive at a tautology: The data prove the need for more data! We think we are creating the system, but the system is also creating us. We build the system, we live in the midst, and we are changed” (89). Concerning a new payroll processing system she has been hired to help develop, she realizes that through her labor “so many entities—employer, software company, bank, IRS—know so much about the simple act of someone getting paid for labor delivered. I'll think about the strange path of a paycheck direct-deposit, how it goes from employer to bank, company to company, while the person being paid is just a blip, the recipient's account a temporary way-station” (188).

Compare these blips as events in software to the processes Knuth implies we ought to trace for just one second, as well as Bogost's unit operations. Fuller now reveals the thought behind the title of his essay. “These blips, these events in software, these processes and regimes that data is subject to and manufactured by, provide flashpoints at which these interrelations, collaborations, and conflicts can be picked out and analyzed for their valences of power, for their manifold capacities of control and production, disturbance and invention” (30). As Manovich describes the no-place of user experience occurring within the space of regulated intellectual property of commercial media performances, blips are also “not merely signifiers of an event, but integral parts of it. . . . They have an implicit politics. . . . There are certain ways in which one is supposed to experience these blips. . . . Your wage statement is the cryptic blip that instantiates the enormous machine of class relations” (31).

Speculative software reflects a design that “refuses to believe the simple, innocent stories that accompany the appearance of these blips” (31). It is characterized by reflexive operations that go where it is not supposed to go, behind the blip, “to make visible the dynamics, structures, regimes, and drives of each of the little events which it connects to” (32). A program like *tcpdump* displays all of the TCP/IP packet headers streaming in and out of your device while performing a simple operation like viewing an account balance. Implicit politics may be revealed in these transactions between Internet addresses by showing the enormous cast of agents involved in what appears to be a singular, personal unit operation, although Fuller's next level of speculative software shakes things up even more.

Not just viewing, but also subjecting blips to unintended forms of connection, speculative software makes “the ready ordering of data, categories, and subjects spasm out of control” (32).

From this deformation of protocol, speculative software then opens to “the havoc of invention” (32). Fuller's text continues with essays describing potential works of speculative software, using Gordan Matta-Clark's architectural “cutting” experiments “to provide the skewed access to the machines that such an investigation requires we can . . . use faults; disturb conventions; exploit idiosyncrasies. . . . Tracking the faults, the severed surfaces, of technology is one way in which this can begin to be done” (44-46). Recalling Bogost's recommendation that software studies be treated materially rather than psychoanalytically, Fuller is suggesting a novel form of analysis that is fitted to the subject matter that is also radically different from the form of media analysis Kittler prescribes. “In any other social context, what appear as protocols because of the arbitrary nature of the machine would be revealed as mannerisms. (Take a fast taxi through the ruined neighborhoods of cyberspace by traveling through emulators of old computers: Punch a hole in the surface of your shiny new machine by loading up the black hole of a 1K ZX81.) . . . Software as an aggregate of very small sensory experiences and devices becomes an engine, not just of connotation, but of transformation” (46-47).[92]

The blips that mark ephemeral manifestations in cyberspace of the dividual cyborg extended mind are entry points to new philosophical understanding. I deliberately called the previous section philosophers of computing technologies to reserve a place for theorists who are all in, close to the machine, who treat the philosophy of software and code as a starting point, basing their insights on original software studies rather than as extensions of a theoretical continuum begun elsewhere in cognitive science or social theory. Wendy Hui Kyong Chun, in

92 Running a Sinclair ZX81 emulator on a modern PC juxtaposes the very ancient and most modern in a way that each artifact may be used to interrogate the other, and additionally—what Fuller leaves out—the experience of the programmer who may have learned working code on the ZX81 as a child.

Programmed Visions: Software and Memory, claims that philosophy is just beginning to note effects of software as a thing on metaphysics, intellectual property, subjectivity, and information. “These changes, brought about by the hardening of software as textual or machinic thing through memory, point toward a profound change in our understanding of what is internal and external, subject and object” (5-6). Indeed, Matthew Fuller wrote this book's Foreword, asserting that “while *Programmed Visions* operates as a sustained introduction to the idea of software, code, and programmability as they work in relation to computation, the book is also a meditation on how this model proliferates, by various means, into systems such as living materials that are in turn understood to be bearers of a form of code that instructs their growth and that can, by further convolution, be read as a print out of the truth of an organism” (vi).

Chun considers the materialization of software as a thing, hardened programming, and memory as a thing, hardened into storage. One important consequence of this transformation is to rethink connections made between the 'undeadness' of new media—their seeming ability to produce new experiences via animated collages from vast recorded archives—as the latest expression of consumer capitalism, and its critique via Horkheimer and Adorno's regressive subjectivity, which I presented in the second chapter. “Although this cycle of the ever-returning and ever-receding new mirrors the economic cycle it facilitates, the undeadness of new media is not a simple consequence of economics; rather, this book argues, this cycle is also related to new media's (undead) logic of programmability. New media proliferates 'programmed visions', which seek to shape and to predict—indeed to embody—a future based on past data” (xii).

Computers are linked to governmentality at the level of their architecture and instrumentality. “By individuating us and also integrating us into a totality, their interfaces offer us a form of mapping, of storing files central to our seemingly sovereign empowered subjectivity. By interacting with these interfaces, we are also mapped: data-driven machine learning algorithms process our collective data traces in order to discover underlying patterns (this process reveals that our computers are now more profound programmers than their human counterparts)" (9). This degree of interrelated mapping exceeds the model of collective intelligence of disciplinary societies that I have been calling Foucault's dust, and makes societies of control possible by orchestrating macro and micro environments so that they appear to be run by a global conspiracy after the fact, or as Chun calls it, by the dream of programmability. "Crucially, this knowledge is also based on a profound ignorance or ambiguity: our computers execute in unforeseen ways, the future opens to the unexpected" (9).

Interfaces are a key component of what Chun calls a neoliberal transformation certifying the dream of programmability to nonexperts by helping to create empowered, productive individuals. Appealing to Ben Shneiderman's work on graphical user interfaces (GUIs), she argues "these interfaces succeed when they move their users from grudging acceptance to feelings of mastery and eagerness" (8). Interfaces also serve as navigational mediators between the visible and invisible, "creating informed individuals who can overcome the chaos of global capitalism by mapping their relation to the totality of the global capitalist system" (8). Finally, "new media empowers individuals by informing them of the future, making new media the future. . . . This future as something that can be bought and sold is linked intimately to the past, to computers as capable of being the future because, based on past data, they shape and predict it" (9).

Before interfaces could be engineered to draw nonexperts into the appeal of new media, the operation of computing machinery had to be easier to visualize in order to assemble extensive systems in the first place. Chun reflects on the historical transformation of pseudocode into source code, into *program* as a noun. “Manfred Broy's software 'pioneers', by making software easier to visualize, not only sought to make the implicit explicit, they also created a system in which the intangible and implicit drives the explicit. They thus obfuscated the machine and the process of execution, making software the end all and be all of computation and putting in place a powerful logic of sourcery that makes source code—which tellingly was first called pseudocode—a fetish” (19).

Sourcery is Chun's term for the fetishization of the visual, human-readable portions of the computing infrastructure as the truly effective means of controlling the inhuman hardware. Therefore, theorists studying code declare it is performative (she cites Galloway); software actually does what it says, fulfilling the ideal of the King's speech in Plato's *Phaedrus* such that “it does not pronounce knowledge or demonstrate it—it transparently pronounces itself” (22). Nuances to the centralization and consolidation of power arguments made by Golumbia, Galloway, Feenberg, and others notwithstanding, in Chun's view both evil-doers like Microsoft and emancipatory free software conceal the *vicissitudes of execution*, the latter more subtly, “by linking freedom and freely accessible source code, amplifies the power of source code both politically and technically” (21). The vicissitudes of execution are all the other material aspects of computer operations required to make the performativity of source code apparent: circuitry, voltages, timing thresholds, clock synchronization, and so on. Additionally, there are the institutional and technical structures that fill out the rest of the environmental, social, and cultural background so that individual instances of computing exist at all.

Showing an example of working PowerPC assembly code to add two numbers, Chun explains that compilation and interpretation are not trivial acts like translating a decimal number into a binary one. “The relationship between executable and higher-level code is not that of mathematical identity but rather logical equivalence, which can involve a leap of faith. . . . Code does not always or automatically do what it says, but it does so in a crafty, speculative manner in which meaning and action are both created” (24). Compilers and interpreters employ heuristics, make multiple passes to optimize an entire set of source code specifications, often changing the internal structure implemented in object code dramatically from what a programmer might expect it to be based on the commands and variables used in the source. Often, the programmer has to make multiple adjustments to the source after the compiler or interpreter flags an error or warning, guiding the final version and somewhat reversing the notion of command and control. That is why “source code is more accurately a *re-source*, rather than a source. Source code becomes the source of an action only after it—or more precisely its executable substitute—expands to include software libraries, after its executable version merges with code burned into silicon chips; and after all these signals are carefully monitored, timed, and rectified” (24-25). A second code snippet shown later is a C++ “hello world” program meant to be easily deciphered. “Programming languages offer the lure of visibility, readability, logical if magical cause and effect” (47). The readability of source code includes embedded natural language in its essential syntax as well as comments; most programming languages exhibit “English-based commands and programming styles designed for comprehensibility” (51).

Referring back to the institutional history of computing, Chun reminds us that “much disciplinary effort has been required to make source code readable as the source” (25). Engineering decisions that were made to implement some functions in hardware, such as Burks, Goldstine, and von Neumann described in over half of the text of their report, were later reversed to allow code to be source rather than circuit, “reducing hardware to memory and thus erasing the existence and possibility of hardware algorithms,” (25) or supplied in a middle ground as microprogrammed libraries permanently burned into read-only memories. “The notion of source code as source coincides with the introduction of alphanumeric [computer] languages. With them, human-written, nonexecutable code becomes source code and the complied code, the object code. Source code thus is arguably symptomatic of human language's tendency to attribute a sovereign source to an action, a subject to a verb” (27).

Chun cites Golumbia and others who connect the sovereign action to Hobbes and the command and control organization of the United States military to software. There is a gendered, military history of computing, shifting from commanding a female computer to commanding a machine. She notes that “software languages draw from a series of imperatives that stem from World War II command and control structures. . . . The Wrens [Women's Royal Naval Service] also (perhaps ironically) called *slaves* by the mathematician and 'founding' computer scientist Alan Turing (a term now embedded within computer systems), were clerks responsible for the mechanical operation of the cryptanalysis machines (the Bombe and then the Colossus)” (29-30). Cybernetics was successful in conflating humans with machines in part because it reduced thought to the common denominator of command languages. “Computation depends on 'Yes, Sir' in response to short declarative sentences and imperatives that are in essence commands” (30).

Interestingly, Grace Kelly Hopper, the famous proponent of programming languages and automatic programming, sought to "put the programmer inside the computer and thus to rehumanize the mathematician: pseudocode was to free the mathematician and her brain from the shackles of programming" (34).[93]

Noting that "this notion of source code as readable as creating some outcome regardless of its machinic execution underlies codework and other creative projects," and "source code as fetish, understood psychoanalytically, embraces this nonteleological potential of source code, for the fetish is a deviation that does not end where it should" (53), Chun proposes the source code fetish creates virtual, authorial subjects, even leading to putative critical acts of revealing sources and connections. Recalling Weizenbaum's dismay over the lack of depth of knowledge in a subject necessary to write programs that involve it, such as the simulation of a Rogerian psychotherapist, Chun goes further, claiming "this erasure of the vicissitudes of execution coincides with the conflation of data with information, of information with knowledge—the assumption that what is most difficult is the capture, rather than the analysis, of data. This erasure of execution through source code as source creates an intentional authorial subject: the computer, the program, or the user, and this source is treated as the source of meaning" (53).

Thus "to know the code is to have a form of 'X-ray vision' that makes the inside and outside coincide, and the act of revealing sources or connections becomes a critical act in and of itself" (53). At the same time, new types of data-driven programming allows machines to write code themselves, producing the black hole of human ignorance to which Kittler alludes the

93 Chun goes on to complicate Golumbia's feminization of protocol aided by his inaccurate description of Hopper as a feminist on account of her powerful role in promoting programming languages and achieving a high rank in the Navy.

machines are going off to answer their own high commands. In this and more everyday ways Chun hints at potential surprises in unknowns that may arise despite programmed vision, like deformative discoveries by McGann. “Embracing software as thing, in theory and in practice, opens us to the ways in which the fact that we cannot know software can be an enabling condition: a way for us to engage the surprises generated by a programmability that, try as it might, cannot entirely prepare us for the future” (54).

However, in everyday use operating systems interpellate users actually and rhetorically; blind faith supplants knowledge of tedious operations performed on behalf of the users that was never there. “Interfaces and operating systems produce users—one and all. Without operating systems there would be no access to hardware; without operating systems there would be no actions, no practices, and thus no user. Each operating systems, in its extramedial advertisements, interpellates a user: it calls it a name, offering it a name or image with which to identify” (66-67). Just as the stored program architecture as proposed by Bush, Goldstine and von Neumann never specified time-sharing and interactive editing, only the light or bell they recommended to signal that a program had halted computation and arrived at a result, it now employs conventions where “computer programs shamelessly use shifters pronouns like my and you—that address you, and everyone else, as a subject” (67).

The conversational familiarity perpetrated by the typical operating system and applications, in the name of user friendliness, helps smooth out the assumed chain from data to information to knowledge. Chun points to a tendency going back at least to Vannevar Bush's famous essay “and in prognoses of the information revolution more generally, there is no difference between access to and understanding of the record, between what would be called, perhaps symptomatically, machine reading and human reading and comprehension, between information and argument, between mapping and understanding. The difficulty supposedly lies in selecting the data, not in reading it, for it is assumed that reading is a trivial act, a simple comprehension of the record's content" (79).

Similar to the fallacy pointed out by Plato at the dawn of writing, a human parallel to ignoring vicissitudes of execution in computers, the aptly named daemonic processes at the heart of modern Unix-like operating systems "transform the computer from a machine run by human operators in batch mode to alive personal machines, which respond to the user's commands. . . . Real-time processes, in other words, make the user the source of the action, but only by orphaning those writing processes without which there could be no user" (89). The modern computer is able to provide interactive interfaces that makes it seem responsive to the users precisely by enacting a multitude of invisible demons. It is worth noting that the key component of the ARPANET's design insisted upon by its principle investigator Wesley Clark was to employ intermediary minicomputers rather than the network host computers to perform the crucial packet switching operations (Abbate 52). Lawrence Roberts, who managed the ARPANET project, called them interface message processors, IMPs, so there was also a demon at the heart of every subnet.

The corollary to programmed visions within the confines of the audiovisual media provided by the computer screen is the transformation of the physical world into what Kitchin and Dodge call code/space. The degree and extent of coding activity permeating everyday life lies on a continuum from coded objects, reliant on software to perform, to coded infrastructures, which are networks of coded objects, to coded processes that "consist of the transactions and flows of digital capta[94] across coded infrastructure," to coded assemblages that "occur where several different coded infrastructures converge, working together in nested systems or in parallel, some using coded processes and others not and become integral to one another over time in producing particular environments, such as automated warehouses, hospitals, transport systems, and supermarkets" (5-7).

Power is relational, arising for code as it does for people out of relationships and interactions. Spaces, both physical and virtual, are where relational power is articulated. For Kitchin and Dodge, software creates and transforms spaces. "The reason why software can modulate the perpetual production of space is because it possesses significant degrees of technicity. This technicity is realized through the process of *transduction*. . . . Transduction is a process of ontogenesis, the making anew of a domain in reiterative and transformative individuations—it is the process by which things transfer from one state to another" (72). Yet software studies to date tend to ignore spaces in which software and people work, focusing on "social information, organization, and regulation, as if people and things exist in time only, with space a mere neutral backdrop" (13). They conceive software as providing a form of governance through automated management of space, as Lessig argues code instantiates law by governing access to intellectual property. They agree with Mackenzie that software possesses a form of secondary agency when it executes itself in automated, automatic, autonomous ways beyond the command and control directives of those on whose behalf it is instituted, but add that "because

94 They prefer the term capta over information or data to designate "units that have been selected and harvested from the sum of all potential data" (5).

software is embedded into objects and systems in often subtle and opaque ways, it largely forms a technological unconscious that is noticed only when it performs incorrectly or fails” (5).

Philosopher David Berry also writes about how such hidden versus visible affordances complicate computational objects; “in a similar way to physical objects, technical devices present the user a certain function . . . but this set of functions (affordances) in a computational device is always a partial offering that may be withheld or remain unperformed” (15). Combining Bernard Stiegler's interpretation of Heidegger, Simondon's concept of a platform, and Wilfred Sellars' phenomenology, Berrry argues the materiality of code as it is tied to phenomena, whether prescriptively creating it or being part of it, must be understood in terms of not only its potentialities as a force, but also as a platform, only ever partially withdrawn—its unreadiness-to-hand. “I want to explore the idea that technology is actually only ever partially forgotten or 'withdrawn', forcing us into a rather strange experience of reliance, but never complete finesse or virtuosity with the technology. . . . I want to develop the argument that we should not underestimate the ability of technology to act not only as a force, but also as a 'platform'. This is the way in which the loose couplings of technologies can be combined, or made concrete (Simondon 1980), such that the technologies, or constellation of technologies act as an environment that we hardly pause to think about” (119-120).

Berry gives the example of Google's instant search that occurs in the midst of entering a query, almost anticipating what you want while subtly guiding you by filling the screen with partial suggestions. “This is the phenomena of 'unreadiness-to-hand' which forces us to re-focus on the equipment, because it frustrates any activity temporarily, that is that the situation requires deliberate attention. . . . The critical question throughout is whether 'computation' is a concept seemingly proper to knowing-that has been projected onto knowing-how/Dasein and therein collapses the distinction between knowing-how and knowing-that hence inducing the substitution of knowing-that for Dasein” (126-127). He presents the as-is situation as a hegemony of computational understanding, featuring decentered, fragmentary subjectivity unified by devices, where “the ontology of the computational is increasingly hegemonic in forming the background presupposition for our understanding the world” (128). As a consequence, “life experiences, then, become processual chains that are recorded and logged through streams of information stored in databanks. . . . In sum, *computer scientists attempt to transform the present-at-hand into the ready-to-hand through the application of computation*” (128-129).

Substituting knowing-that for Dasein reiterates in the technical, philosophical language of Heidegger a common theme presented throughout this dissertation: it is latent in Bauerlein's dumbest generation, Winner's absent mind, Weizenbaum's compulsive programmer, Turkle's robotic moment, Golumbia's presentism, and Chun's programmed visions. The switching costs of unready-to-hand technological comportment of running software affects contemporary subjectivity. Berry's solution is to develop Lyotard's distracted consciousness stream alongside Deleuze and Guatarri's schizophrenic, to suggest a computational way-of-being, perhaps as Turing super-cognition and Clark extended cognition, enabling exteriorization of cognition and reflexivity, while at the same time being careful to avoid 'screen essentialism'. He argues the loose coupling network layers are another good example of loosely independent connections thought in terms of Heidegger *Gelassenhei*t, where a relaxed type of relation to technical devices “points towards the possibility of a relationship with technology that is not built of the will to power by virtual of the impossibility of control in a system that exceeds the comprehension of a human subject," but rather one of serenity, letting go, with abstraction taking place all the way down (140).

Berry develops an ethic for the philosophy of software that can be summarized as being a good stream. "The real-time stream is not just an empirical object; it also serves as a technological imaginary, and as such points the direction of travel for new computational devices and experiences. . . . The new streams constitute a new kind of public, one that is ephemeral and constantly changing, but which modulates and represents a kind of reflexive aggregate of what we might think of as a stream-based publicness which we might call *riparian-publicity*. Here, I use riparian to refer to the act of watching the flow of the stream go by" (142-145). Restructuring post-human subjectivity as riding on top of networks of computationally-based technical devices is the key point of the book, adding Lyotard's postmodern fables, Massumi's affective facts, and Serres' parasite to the material phenomenology of code. "This notion of a restructured subjectivity is nicely captured by Lucas (2010) when he describes the experience of dreaming about programming. . . . This is the logic of computer code, where thinking in terms of computational processes, as processual streams, is the everyday experience of the programmer, and concordantly, is inscribed on the programmer's mind and body" (149).[95] This materiality of code becomes inscribed in programmers through long habituation, internalized to point of dreaming, whereas the materiality of software becomes inscribed in users, for example as multitasking synaptogenesis. This conception severely challenges the liberal humanist

95 In *Dreaming in Code* Rosenberg attributes the term to Jaron Lanier, who is alluding to future forms of interpersonal, machine-mediated communication that invoke coded objects in place of linguistic rhetorical forms.

individual; bounded rationality is replaced by extended cognition, thus appealing to visual rhetorics comprehending so-called Big Data, which is Manovich's current preoccupation.

This ethic of being a good stream, invoking Serres' parasitic subjectivity, although requiring thoughtful comportment to computer technologies, does not endorse outright learning and practicing programming as itself critically important the way literacy did for prior generations, or as a substantial component of humanities scholarship's intellectual labor. “The parasite is used not as a moral category, but in connection with an actor's strategic activities to understand and manipulate the properties of a network. . . . The question of who this subject 'eats next to', is perhaps reflected in the way in which streams pass through other streams, consumed and consuming, but also in the recorded moments and experiences of subjects who remediate their everyday lives. This computational circulation, mediated through real-time streams, offers speculative possibilities for exploring what we might call parasitic subjectivity” (171). Berry's final thought is this Serres-inspired parasite subjectivity in symbiotic relationship to the enormous host machinery generating the digital standing reserve: “Lyotard's comment to the streams that flow through our postmodern cultural economies seems as untimely as ever: true streams are subterranean, they stream slowly beneath the ground, they make headwaters and springs. You can't know where they'll surface. And their speed is unknown" (171). I see a fundamental flaw in this image because passing through underground cavities to surface waters involves reduction to lower levels of the network ontology before resurfacing. The difference between the programmer's dreaming in code and the everyday user's reliance on cultural software mediations is my intuitive grasp of the as-is collective intelligence problem, and requires us to move from software studies to code studies to articulate it fully.

Code Studies

Berry concludes his philosophy of software with the ethical imperative of being a good stream. Like numerous other thinkers who have contributed to my theoretical framework, he provides many insightful, smooth lines of flight on which further study can travel, yet also leaves me dissatisfied with a lack of technical, programmer-savvy options in the spirit of McGann's call for poiesis-as-theory, what Applen and McDaniel call theorist-practitioners, and Hayles puts under the Big Humanities tent. I have already pointed out with others the ambivalence toward technology in Deleuze's individual writings that seems to offset the machinic enthusiasm he shares with Guattari in *A Thousand Plateaus* that ought to have inspired multiple generations of digital humanists to pick up soldering irons and build things. The lure of fetishization of source code Chun proposes may also contribute to a reasoned turning-away from programming and code studies as an explicit, scholarly activity versus practicing critique at a safe distance.

Yet code studies—and platform studies, which will be articulated in the next section—complete the lower levels of a diagram of ontic layers of network phenomena that cultural software and technology criticism ride atop. No printed, scholarly text makes this point better than Montfort et. al.'s 2013 work *10 PRINT CHR$(205.5+RND(1)); : GOTO 10*, whose title is a one line computer program that, when typed into a Commodore 64 and run, generates an endless, maze-like output. In it they declare “Critical Code Studies is a set of methodologies for the exegesis of code. Working together with platform studies, software studies, and media archeology and forensics, critical code studies uses the source code as a means of entering into discussion about the technological objects in its fullest context. CCS considers authorship, design process, function, funding, circulation of the code, programming languages and paradigms, and coding conventions" (6-7).

Mark Marino, one of this work's many contributors, posted a CCS manifesto in 2006 declaring "an approach that applies critical hermeneutics to the interpretation of computer code, program architecture, and documentation within a socio-historical context" (np). Its core methodology "follows the work of Critical Legal Studies, in that its practitioners apply critical theory to a functional document (legal document or computer program) to explicate meaning in excess of the document's functionality, critiquing more than merely aesthetics and efficiency" (np). Stressing meaning, implication, and connotation, thought not just in terms of a self-contained system of meaning but with respect to the broader social contexts, "practitioners may critique the larger human and computer systems, from the level of the computer to the level of the society in which these code objects circulate and exert influence" (np). Thus, Marino's earlier articulation of CCS is somewhat more focused than I wish to be here.

I am positioning code studies alongside software studies as it has been articulated in the previous section, encompassing programming languages and programming styles, so that aesthetics and efficiency remain in scope along with Montfort et. al.'s aforementioned "authorship, design process, function, funding, circulation of the code, programming languages and paradigms, and coding conventions" to foster "discussion about the technological objects in its fullest context." Marino does take a stand against John Cayley and others who insist the code to be studied must be executable. On the contrary, everything surrounding the code and resembling code can be interpreted. "The code, the documentation, the comments, the structures all will be open to interpretation. . . . Within CCS, if code is part of the program or a paratext (understood broadly), it contributes to meaning. . . . Within the code, there will be the actual symbols but also, more broadly, procedures, structures, and gestures” (np). By engaging at the level of source code and other artifacts of the social production of software—and taking seriously the vicissitudes of execution that the platform level considered in the next section informs—Richard Johnson's two key insights about popular culture studies are realized for computing: “The key insight for me, is that narratives or images always imply or construct a position or positions from which they are to be read or viewed. . . . If we add to this, the argument that certain kinds of texts ('realism') naturalize the means by which positioning is achieved, we have a dual insight of great force. The particular promise is to render processes hitherto unconsciously suffered (and enjoyed) open to explicit analysis” (66). First, what is code?

The *Software Studies Lexicon* entry “Code,” written by Kittler, bears the subtitle “(or, How You Can Write Something Differently),” alluding to code's historical connection to encryption, although characteristic of every transmission medium. He claims that “codes became conceivable and feasibly only after true alphabets, as opposed to mere ideograms or logograms,” and appeals to Suetonius's *Lives of the Caesars*, which “recounts discovering encrypted letters among the personal files left behind by both the divine Caesar and the divine Augustus,” to suggest “the basis on which command, code, and communications technology coincided was the Empire” (40-41). Codes remained in use for secret messages for centuries, but Morse Universal Code Condensers were the first deliberately designed for efficiently transmitting information, so that “a system of writing had been optimized according to technical criteria – that is, with no regard to semantics,” replacing enciphering and deciphering with encoding and decoding (43). By the time Turing was thinking about code, “the twentieth century thus turned a thoroughly capitalist money-saving device called 'code condenser' into highest mathematical stringency" (43). Importantly for Kittler's account, "Turing himself raised the question of the purpose for which computers were actually created, and initially stated as the primary goal the decoding of plain human language," but (quoting from "Intelligent Machinery") "to depend rather too much on sense organs and locomotion to be feasible. . . . the subject matter of cryptography is very easily dealt with by discrete machinery, physics not so easily" (44).

As Suchman found decades later, knowledge of situated actions in the environment are essential to artificial intelligence, pushing the field in the direction of machine learning and automated, data-driven programming, which, Kittler maintains, results in "this new and adaptable environmental knowledge in robots would remain obscure and hidden to the programmers who started them up with initial codes" (45). He pessimistically concludes "either we write code that in the manner of natural constants reveals the determinations of the matter itself, but at the same time pay the price of millions of lines of code and billions of dollars for digital hardware; or else we leave the task up to the machines that derive code from their own environment, although we then cannot read – that is to say: articulate – this code" (46).

Whether written by humans or machines, code *functions*. The potential for automated, autonomous, automatic executability of code differentiates it from other texts and semiotic systems, while sharing personal and cultural significance with other types of texts. Montfort et. al. consider code as a cultural resource, as well, "not trivial and only instrumental, but bound up in social change, aesthetic projects, and the relationship of people to computers. Instead of being dismissed as cryptic and irrelevant to human concerns such as art and user experience, code should be valued as text with machine and human meanings, something produced and operating within culture” (8).

Berry defines source code as “the textual form of programming code that is edited by computer programmers,” which presents the challenge of understanding code as a textual artifact (29). A second aspect of code is “something in process, as it executes or 'runs' on a computer,” distinct from its textual form (29). This characterization has a historical period associated with it—source code as textual form that is compiled into object code, or interpreted on the fly during execution in its native format as the machine code of a specific computer architecture—that has been well understood for a number of decades, and remains the status quo, although Chun, Kittler, Kurzweil, and others predict data-driven automatic programming by machines will eventually outpace the production of source code by humans. Chun goes so far as to discern a difference between coding and programming, too, dating back to von Neumann's theories of general automata. Turing, she notes, “did not refer to short or complete codes, but rather to instructions and tables to be mechanically—meaning faithfully—followed” (166). Von Neumann, on the other hand, distinguished between short codes and complete codes. The former referred to the representation of mathematical expressions, whereas the latter were strategies, “a list of instructions to be followed under various conditions,” ideally such that an automaton can replicate itself (165).

Chun claims that “in a remarkably circular route, von Neumann establishes the possibilities of source code as logos: as something iterable and universal. Word becomes action becomes word becomes the alpha and omega of computation” (167). She argues this equivalence of describing and producing an object, substituting word with deed, is linked to the equivalence of access and comprehension, setting up programmed visions. The question remains who or what transforms descriptions into instructions, and makes humans and automata indistinguishable.[96] That, of course, becomes the programmer, but arguably one who approximates the behavior of ideal players in game theory, which von Neumann also had a role in defining: “[Gregory] Bateson is absolutely correct in his assessment: in outlining such a comprehensive version of a strategy, game theory assumes a player who could only be—or later would become—an automaton. . . . A strategy is something an automaton—or more properly a programmer—working non-'interactively' with a computer has” (166). However, strategies articulated as assembly languages were invariably tied to the particular machine architecture for which they were written. For a long time, the level above assembly language was referred to as pseudocode. So-called higher-level programming languages “enabled the separation of instruction from machine, of imperative from action, a move that fostered the change in the name of source code itself from pseudo to source” (41).

Knuth and Pardo document the early history of programming languages from von Neumann's time through the mid 1950s, but are careful to note that many of them had very small user bases and invariably were tied to particular machines. As they note, “the best languages we

96 She bases her argument for memory, stored instructions, and primacy of software on von Neumann's use of McCulloch and Pitts' neuron model rather than Shannon's communication model, a theme Hayles explores in *How We Became Posthuman*. To make the model work, von Neumann hypothesized the existence of a biological memory organ, which had an equally powerful influence on cognitive science and AI research. “Through this memory organ, von Neumann would erase the difference between storage and memory, and also open up a different relationship between man and machine, one that would incorporate instructions as a form of heredity into the machine, making software fundamental” (157).

shall encounter are, of course, very primitive by today's standards, but they were good enough to touch off an explosive growth in language development" (2). Like Chun they note Turing's incomplete prototype machine language, "not having a built-in arithmetic capability, and he defined a complex program by giving what amounts to macro-expansions of open subroutines," as well as Alonzo Church's lambda-notation.

To Campbell-Kelly, the software historian, programming languages came into being when their existence was ratified by its primary consumers within the budding military-industrial-educational complex. "The crucial breakthrough in programmer productivity was the development of programming languages in the second half of the 1950s. In a programming language, a single line of code could generate several machine instructions, improving productivity by a factor of 5 or 10" (34). FORTRAN, invented for the IBM 704 at IBM's Technical Computing Bureau under the direction of 29-year-old John Backus, received immediate and ecstatic responses from its users. "Backus and his team refined the system in response to feedback from users and released a new version in 1959. FORTRAN II contained 50,000 lines of code and took 50 programmer-years to develop" (35).

Rosenberg claims that source code as a distinct type of text in fact arose with FORTRAN: "laborious sequences of assembly language procedures were summarized in brief commands. The human programmer would write a sequence of these commands—source code; then a kind of uberprogram running on the computer called a compiler would translate those commands into object code in the machine's own language" (67).[97] Network effects popularized FORTRAN

97 Rosenberg recounts the origin of source code to accompany the claim by Guido Van Rossum, inventor of the language Python, that "a Python program can usually accomplish the same task as a C or C++ program using three, five, or even ten times less code" (71).

more so than help from IBM marketing. The United States Department of Defense then pushed the industry to agree on a standard business language, from which COBOL arose. According to Campbell-Kelly, FORTRAN and COBOL garnered two-thirds of application programming activity for the next twenty years. We encountered Kemeny and Kurtz in the first chapter, who together developed and promoted BASIC as a programming language for college students, although C and Unix-like operating systems became the preferred language for universities for many years. The field is broad today, with C++, Java, C#, Perl, Python, and PHP in widespread use, although most histories of programming languages point out that more idiosyncratic dialects flourished in the early days of computing than the present.

Returning to the *Software Studies Lexicon*, the entry “Source Code” by Joasia Krysa and Grzesiek Sedek starts with the first page of a C program called “recipe” for the Barszcz project, featuring comments, preprocessor directives known as 'includes' and 'defines', and one compact function demonstrating, for those who know it, the elegance and oddity of C to use pointers (`**take` and `**ingr` in this example) to both variables and functions, illustrating the legacy of Babbage, von Neumann, and Remington Rand's UNIVAC as “one of the first machines to combine electronic computation with a stored program and capable of operating on its own instructions as data” (238).[98]

98 I preserve the formatting of this C code snippet besides adding the three dot ellipsis to designate a region of omitted lines. Tanaka-Ishii goes into specific details about her rationale for typographic conventions (typewriter face for working code, italics for mathematical notations, single quotes for terms and phrases, double quotes for inline quotes from other references (9)), but in most humanities texts that cite source code there is little uniformity beyond a signal like italics, boldface, or a different font to differentiate source code from literary text.

```
/ Barszcz C recipe
* string based cooking
* Copyleft (C) 2006 Dennis “Jaromil” Rojo
#include <stdio.h>
#define ingredient char
...
ingredient **take(int quantity, ingredient **ingr) {
      int c;
      int len = strlen(ingr) +10;
      ingredient = malloc ( (quantity+1) * sizeof(*ingredient));
      for(c = 0; c < quantity; c++)
            ingredient[c] = malloc(len * sizeof(ingredient));
      ingredient[c+1] = NULL;
      return ingredient;
}
```

This code snippet, as such a small but illustrative portion of an overall source code text is often called when displayed to make a technical point, just as academics cite brief passages to substantiate their scholarly insights, leads into Donald Knuth's equating programming to “recipes in a cookbook as a set of instructions to follow” in his famous book *The Art of Computer Programming* (236). Besides including programmers' comments that do not have a role in execution but record authorship, licensing, descriptions, and instructions, “the importance of source code is that any modifications (improvements, optimizations, customizing, or fixes) are not carried out on compiled binary code (object code or machine code) but on the source code itself” (237).

Source code is where the programmer's influence can be exerted, versus the vicissitudes of execution that resemble the oven baking the cook's carefully measured and arranged ingredients. Krysa and Sedek refer to its Copyleft copyright notice as emblematic of free, open source software, noting that “an online repository and a platform for presenting and sharing barszcz soup recipes in the form of source code written in a number of programming languages, the [*Barszcz.net*] project brings together cooking recipes and source code in a literal sense. In a wider cultural context, this exemplifies a general way of thinking about source code as an open model for creative practice; it can be used to encourage collaboration and further development of existing work on the level of contribution, manipulation, and recombination, and can be released under the same or similar licenses in the public domain” (240). They refer to a number of websites featuring code that may be of interest to digital humanists. “Source code can be considered to have aesthetic properties; it can be displayed and viewed. It can be seen as not only as a recipe for an artwork that is on public display but as the artwork itself as an expressive artistic form that can be curated and exhibited or otherwise circulated” (239).

Berry distinguishes code as the “static textual form” and software as “processual operating form” of what is collectively called software, and contends that “we must also not be afraid of using other technical devices to mediate our access to the code . . . we can use devices to open any level of code to view (e.g. debuggers, IDEs, oscilloscopes)” (31-33). Consequently, the appearance of code in a print document assumes some liberties have already been taken to acquire it and format it for display. Moreover, the snippets may need to be read from a particular absolutist perspective that seems counter to the sensitivity to context that has been a theme of much of the prior research. Berry makes the qualification that “in a Clausewitzian sense, I want to think through the notion of 'absolute' code, rather than 'real' code, where real code would reflect the contingency and uncertainty of programming in particular locations. Absolute code would then be thinking in terms of both the grammar of code itself, but also trying to capture how programmers think algorithmically, or better, computationally” (33).

Recalling Weinberg's discussion about how to read code, Berry confirms “programmers have tended to need to continually use this hermeneutic circle of understanding 'the parts', 'the whole' and the 'parts in terms of the whole' in order to understand and write code (and communicate with each other), and this is striking in its similarity to literary creativity, to which it is sometimes compared" (34). Importantly, the fact that code work occurs in the mediated environment of software engineered for developing software means the appearance of code snippets on a printed page is actually a skeumorph of earlier times. "This means that software is mediating the relationship with code and the writing and running of it. When we study code we need to be attentive to this double mediation and be prepared to include a wider variety of materials in our analysis of code than just the textual files that have traditionally been understood as code" (37). Berry prefers not to limit the phenomenology of software to these presentations of source code; "code needs to be approached in its multiplicity, that is, as a literature, a mechanism, a spatial form (organization), and as a repository of social norms, values, patterns and processes" (36).

No form of code is more important than its run time operation, since that is why it is developed in the first place. "Code is processual, and keeping in mind its execution and agentic form is crucial to understanding the way in which it is able to both structure the world and continue to act upon it" (38). His recommendation is to think about how code executes by sequential ticks through each statement, with attention to the illusion of concurrency. "This means that code runs sequentially (even in parallel systems the separate threads run in an extremely pedestrian fashion), and therefore its repetitions and ordering processes are uniquely laid out as stepping stones for us to follow through the code, but in action it can run millions of times faster than we can think – and in doing so introduce qualitative changes that we may miss if we focus only on the level of the textual" (38). His phenomenology of computation is performed by such reverse engineering of discretizations, in addition to recognizing political economy, property relations, breakdowns, and moral depreciation concretized in code and software systems. For instance, using programming tools like debuggers, “by slowing down or even forcing the program to execute step-by-step under the control of the programmer the branches, loops and statements can be followed each in turn in order to ensure the code functions as desired” (38). Programmers are used to these multiple views of code, including stepwise tracing of the execution of object code that more or less maps onto specific source code statements, depending upon how it is compiled or interpreted.

Understanding that code is manufactured directs us to the political economy of software, where “code needs to be thought of as an essentially unfinished project, a continually updated, edited and reconstructed piece of machinery,” that is also “treated as a form of property in as much as it is subject to intellectual property rights, like copyright” (39-40). Moreover, software continually breaks down; much never reaches working state, much is never used, much remains hidden in large code repositories. “Software, therefore, always has the possibility of instability being generated from within its different logics. Even when it is functioning, code is not always 'rational'” (42).[99] The implicit complexity through distributed authorship over many revisions means it is likely that nobody comprehends all of any given application. The reality that much code never leaves the confines of private businesses, and even less code sees the light of public dissemination, hints at the software development lifecycle (SDLC). “It is created as code, it is developed whilst it is still productive, and slowly it grows old and decays, what we might call the

99 Berry refers to the research by Adrian MacKenzie on professional software development practices and the business of cutting code. Parallels to the earlier work by Weinberg and Kraft are evident.

moral depreciation of code. . . . So software too can suffer a kind of death, its traces only found in discarded diskettes, the memories of the retired programmers, their notebooks, and personal collections of code printouts and manuals” (42-43).

The sloppy, contestable, always partially concealed view of code Berry presents from the perspective of professional software development differs markedly from the clear, absolutist snippets presented piecemeal to illustrate coding principles. “Nonetheless we should be clear that the ontology of code is specifiable, indeed, programmers already know what code is, qua code” (43).

Phenomenologically derived ontological characteristics from the experiences of programmers are based on habituation, structural constraints, and shared knowledge. “These methods of habit become deeply ingrained in the way in which a programmer will tackle a computing problem and are demonstrated by the way in which programmers often become attached to particular programming languages, shells, editing environments or computer platforms (e.g. Unix). . . . This has been reinforced by marketing efforts which use 'fear, uncertainty, and doubt' (FUD) to encourage customer loyalty” (44). Structural constraints imposed by Integrated Development Environments (IDEs) and compilers require “a very high degree of proof-reading ability in order to ensure the correctness of the program under development . . . a form of prescribed literate programming,” (44-45) or a form of automatic programming when the IDE provides prompts and suggestions, such as 'Intellisense' familiar to Microsoft Visual Studio users.

Code may compile but not work, or contain syntax errors, deprecated functions, or other flaws so it will not compile or run despite being syntactically well-formed. Consequently, there is a big difference in practices between collaborative and personal projects. Organizations that maintain code “can be an extremely social activity with shared norms as to how the code should be laid out, what are the acceptable and unacceptable ways of writing a program” (45). Code reviews, partner programming, and the distributed mechanisms of “free/libre and open source software (FLOSS) which allows anyone to download and read the code, all act to promote readability and excellence in the programming” (45).

The ontology of code includes metaphorical aspects as well. Grand narratives and cultural tropes are related to metaphorical code, such as engine, image, communication medium, and container. Berry proposes Weberian ideal-types of analytical categories to build a grammar of code: data at the basic level, code as the mechanism operating on data, delegated (source) manipulated by programmers, prescriptive (software) compiled and translated into machine operations, commentary offering “a hermeneutic and historical record in commentary code in addition to the processing capabilities of the explicitly delegated code within the file,” and finally critical code (52-54). Berry's critical code is different than Marino's. “This is code that is written to open up existing closed forms of proprietary computer code, providing users the ability to read/hack existing digital streams and hence to unpack the digital data structure” (53). An example is software that examines TCP/IP data streams like `tcpdump`, although it is essential that the source code be available “for inspection to check the delegated processing that the code undertakes” (54). I have been calling this criterion *epistemological transparency*.

While Berry makes the best attempt of any theorist I have encountered to propose an actionable philosophical approach to align ourselves with the plane of immanent connections jointly shared by human and machine networks that is thoroughly grounded in software and code studies, *The Semiotics of Programming* by Kumiko Tanaka-Ishii exemplifies how theories from other disciplines can frame and shape our understanding of computers and their limits, roles, or functions in society, and conversely, how that refined understanding of computer technology can be used to revisit classic questions in the humanities. A theme I introduced at the outset of this dissertation is how tension between the spoken and written word alludes to reasons for discounting machine languages and excluding them from study in the humanities, evident in Ong and Derrida, but ultimately referring back to Saussure's inauguration of semiotics, where he states "the object of study in linguistics is not a combination of the written word and the spoken word. The spoken word alone constitutes that object. But the written word is so intimately connected with the spoken word it represents that it manages to usurp the principal role. . . . A literary language enhances even more the unwarranted importance accord to writing" (24-26).

Rather than turning away from literary languages, if we admit that the models of signs, kinds of signs, and systems of signs evident in programming languages reflect back on natural languages, Tanaka-Ishii makes the claim that "understanding the semiotic problems in programming languages leads us to formally reconsider the essential problems of signs. Such reconsideration of the fundamentals of semiotics could ultimately lead to an improved and renewed understanding of human signs as well" (4). While individual chapters delve into specific topics in semiotics and their correlates in computer science, the overall theme is reconsidering reflexivity as the essential property of sign systems. Reflexivity is taken for granted in natural languages, although various paradoxes alluding to the inherent risk of unintelligibility are well known topics in philosophical logic. "With artificial languages, however, it is necessary to design the border of significance and insignificance and thus their consideration will serve for highlighting the premise underlying signs and sign systems" (1).

We can understand signs by looking at machines for intersections, as Derrida did with writing and speech. The border of significance made explicit in the design of artificial languages—specifically programming languages, "and it is not too far-fetched to say that the history of programming language development is the quest for a proper handling of reflexivity"—may lead to new insights for semiotics in general or clarify existing, long held positions (2). Tanaka-Ishii appeals to this common test bed of sign systems due to the extent humanities disciplines treat humanity as discursive, a point made by Hayles in her study of cybernetics, Golumbia in his critique of computationalism, and others repeat. Instead of questioning this premise, Tanaka-Ishii proposes we run with it, stressing instead "the difference between computer signs and human signs lies in their differing capability to handle reflexivity, which informs both the potential and the limitations of the two sign systems. While people do get puzzled, they seldom become uncontrollable because of one self-reference" (3).

Like the way Kittler problematizes media studies, semiotic studies are seldom delineated from their expressive symbolic systems, but they can be by using programming languages as a test bed. As Bolter argues in *Writing Space*, "The electronic writing space seems to be not a metaphor for signification, but rather a technology of signification. Signs in the computer do precisely what students of semiotics have been claiming for their signs for more than a century as they generate text automatically" (176). Tanaka-Ishii adds that, since humans do not think in machine language, there is a fundamental separation, implying the texts that they automatically generate possess a sort of authenticity absent in contrivances that play on typography. At the same time, because computer languages "are truly formal, external, and complete in scope," and "interpretation is performed mechanically, it is explicit, well formed, and rigorous," as well as being “the only existing large-scale sign system with an explicit, fully characterized interpreter external to the human interpretive system,” (4) they seem an appropriate place to do work in semiotics. Indeed, Tanaka-Ishii suggests that object-oriented programming principles are latent in earlier semiotic theory; “the technological development of programming languages has thus been a rediscovery of ways to exploit the nature of signs that had already been present in human thought,” and “the problems existing in sign systems generally also appear in programming languages” (5).

Coming full circle, technological development inspires humanities study, as others have claimed computer technologies are applied poststructuralism and postmodernism. This history of the semiotics of computing starts with Zemanek in 1966 emphasizing the importance of semiotic analysis of programming languages, but actual studies modeling computer signs did not appear until the 1990s with Andersen, Holmquvist, and Jensen, after which a number of journals began publishing on the subject (6). She also notes Floridi's wide-ranging philosophy of information and de Souza's application of semiotics to human-computer interaction. Her work makes no claim of extending theories of programming languages; “I refer only to theories and concepts already extant within the computer programming domain and merely utilize them for semiotic analysis: since a programming language is well-formed and rigorous, the relevant theory is fundamentally clear” (8). Thus most chapters use Haskell and Java to demonstrate the arguments from the functional (procedural) and object-oriented perspectives. She suggests ambitious humanities readers may be able to grasp program operations judged simple for those trained in computer science; practicing programmers may be in the middle, depending on their formal education in computer science and mathematical logic.

As an introduction to working code, she provides a fifteen line Haskell program, and a twenty-seven line Java program that both calculate the area of rectangles, circles, and ellipses. In describing them, she identifies four types of signs appearing in computer programs: literals, operators, reserved words, and identifiers. “The first three kinds of signs are defined within the language system, and programmers merely utilize them. . . . In contrast, identifiers are defined by the programmer, and a program consists of hierarchical blocks of identifier definitions and uses” (17). Identifiers were literally memory addresses in early programming languages and low-level assembly language, whereas “today's identifiers are abstract representations of memory addresses in the form of signs” (17). Computer signs, for the sake of her arguments, are identifiers in programs; “most other language-specific signs are defined as identifiers in the metalevel language that describes the language” (18). These identifiers may represent data values, functions, or complex structures consisting of both—marking the boundary between traditional procedural languages and object-oriented ones. Overall, this characterization implies the stored program architecture, and an implicit ontology of programs as hierarchical blocks similar to the OHCO theory of textuality.

In addition to focusing on identifiers within programming languages as the significant computer signs for semiotic analysis, Tanaki-Ishii employs a *pansemiotic view*. Borrowing from Charles Sanders Peirce who developed the term to suggest entities exterior to the mind can only be grasped through representation by signs, “setting the interpretation level at the programming language level means considering the interpretation of signs *within* the semiotic system. It does not require external entities such as the physical objects that a program represents,” nor does it deny the existence of external entities; “the computing world is a rare case in which the basic premise of pansemiotic philosophy holds" (20).[100] From the pansemiotic perspective, the semantic levels of computer identifiers are the hardware level, programming language level, and natural language level. "An identifier therefore represents both an address and a value in bits at the hardware level. . . . This semantics at the computer hardware level is now becoming more the domain of professionals who build compilers and optimizers, whereas programmers tend to handle programs only at the higher levels of programming languages and natural languages" (18).[101]

At the programming language level all identifiers are defined and then used, whether they are single letters like `r` or strings of letters like `Shape`. This level also distinguishes *types* to limit the kind of data or function an identifier can represent, as well as the kinds of expressions in which it can be used. Most contemporary languages supply a number of native data types like integers, floating point numbers, characters, strings, and support the declaration of user-defined data types that may include functions as well. Besides the layer of type, programming languages may also specify a layer of *address*, when an identifier represents a memory address directly rather than the type of contents of a memory address. The natural language level includes free form comments surrounding program code proper that are intended for human consumption, and more importantly "the interpretation of identifiers that are apparently formed of natural language terms . . . that give clues as to the intent of a program" (20). Indeed, "programmers are trained to

100 Tanaka-Ishii notes Saussure and Augustine as both proposing that mind implies signs, but could also invoke Clark's parity principle to allow the study to be abstracted from questions of the nature of intelligence.

101 This admission counters Kittler's claim that there is no software.

choose and design meaningful identifiers from a natural language viewpoint," and these identifiers "are considered subject to normal semiotic analysis of terms in natural language" (20).

On the one hand, this is the point of departure for most humanities-based code studies, which point out the hegemony of English, use of masculine or feminine pronouns, implicit colonial or moronic characterizations of users, and so on, in comments and natural language identifiers. Rosenberg brings up code studies of profanity in the Linux kernel source code and the leaked Windows 2000 source code, noting that "comments sometimes serve not just as explanatory notes but as emotional release valves. Despite decades of dabbling with notions of automatic programming and software engineering, making software is still painful. Anguished programmers sometimes just need to say fuck" and "among the surprises they found were many comments in which the Microsoft programmers berated themselves, their tools, their colleagues, and their products" (307-309). Berry presents many more insightful studies involving the leaked Microsoft code, climate research code, and especially the extended analysis of the implied subject positions of the users in the VoteBox voting machinery. "This idealized voter constantly seeps into the source code in a number of interesting ways, captured in the shorthand used in the commentary code and documentation. Although here we do not have the space to go into the interesting gender assumptions that are being made, they demonstrate how programming commentary is important to analyze and take into account as part of the assemblage that makes up running software systems" (114).

Tanaka-Ishii's focus, on the other hand, is admirable for resisting the temptation to take such tangential paths, and instead tackle big, abstract problems in semiotics that have puzzled philosophers for centuries. Moreover, it requires sustained attention to the semantic level of the programming languages, which is often quickly passed over to engage in the natural language level. When she takes on dyadic and triadic sign models from Augustine and Greek philosophy, she teases out their correspondences between sign models of Saussure's relatum and signified dyad (completed with the excluded thing), to Perice's signifier, object, and interpretant triad as articulated by Nöth and Eco. She proposes the new hypothesis that Saussure's signified corresponds to Peirce's immediate object, and interpretant in language system outside sign model appearing as difference in use. That is, “the dimension of reference or sense is not *ignored* by Saussure; rather, the interpretant is simply situated outside the sign model, appearing as difference in use” (34). Then she tests these sign models with the aforementioned programming paradigms: where is the common area function located with respect to definition of shape?

The answer introduces the basic distinctions between functional and object-oriented programming, where data definition remains minimal in the former, maximal in the latter. Her analysis finds that functional programs employ dyadic identifiers exclusively, while dyadic and triadic identifiers appear in object-oriented programs. “A sign in the dyadic model has a signifier and a signified. Because all dyadic identifiers consist of a name and its content, the name is likely to correspond to the signifier and the content to the signified” (36). Dyadic identifiers acquire meaning from use located external to their calling context in the functional paradigm, relating to Saussure's model. “A sign in the triadic model has a representamen, an object, and an interpretant. Since all triadic identifiers in the object-oriented paradigm consist of a name, data, and functionalities, these lend themselves respectively to comparison with the relata of the triadic sign model” (37). A class has information about its functionality, so meaning and use are partially built in, as Peirce's model suggests. “The concepts of icon/index/symbol and denotation/connotation plus object language/metalanguage correspond within the context of application to programming languages. Similar concepts have been developed within both the dyadic and the triadic frameworks, and the resulting correspondences validate each framework" (107).

Subsequent chapters in *The Semiotics of Programming* ponder the marriage of signifier and signified via a Chomskian recursive definition of a grammar by using rewrite rules that bases lambda calculus; being and doing in the progression of object-oriented languages from class emphasis to abstract data types and interfaces; the semiotic ambiguity of identifiers in Barthes' sign studies and Hjelmslev's glossematics through call by value versus call by reference, and so on. The final chapter "Reflexivity and Evolution" reconnects to broader themes that have been addressed throughout this dissertation in posing the collective intelligence problems of the as-is situation, and the responses proposed by various philosophers of computing technologies. I am not doing justice to the subtly of Tanaka-Ishii's analyses with this quick summary. My point is that a theorist familiar with both computer science and the broad representation of philosophical traditions she invokes would be convinced that she has found a new place to enter and do work in the philosophy of computing.

As brilliant as the analyses conducted by Tanaka-Ishii may be, they nonetheless operate in the rarefied environment of carefully crafted coding examples for the topic under consideration, the semiotics of programming. Her position, in effect, presupposes one has already cultivated an ability to study code and interpret it. In *Electric Language*, Michael Heim argues that language contains essential, systematic ambiguity. "Far from being a tool of thought, language, or logos, is the element of thought. Heraclitus insists: What we think about language is implicated in how we think in language; and, alternatively, the manner in which we think in and through language entails a certain stage or degree in our understanding of language" (35). He fears that word processing seeks to make all utterances well formed and unambiguous like program source code. While this fear may be realized by the programmed visions Chun details, the foregoing discussions support the often provisional nature of source code, in which it does not fully materialize as working code until after the fact, after programmers have massaged their intuitive plans about what it is supposed to do into situated actions. In the entry on "Obfuscated Code" by Nick Montfort in the *Software Studies* lexicon, he remarks that "a program may be clear enough to a human reader but may have a bug in it that causes it not to run, or a program may work perfectly well but be hard to understand. Writers of obfuscated code strive to achieve the latter, crafting programs so that the gap between human meaning and program semantics gives aesthetic pleasure" (194).[102]

The example Montfort gives is the `10 PRINT CHR$(109+RND(1)*2);: GOTO 10` one-liner, whose cousin becomes the subject of an entire book he and a number of collaborators published a few years later. Confusion between data, code, and comment, also discussed by Tanak-Ishii, results from flaws or ambiguities in a language's specification, particularly C and Perl. "By showing how much room there is to program in perplexing ways—and yet accomplishing astounding results at the same time—obfuscated C programs comment on particular aspects of that language, especially its flexible and dangerous facilities for pointer

102 Monfort includes Alfred Jarry's Pataphysics, "the science of imaginary solutions" as a predecessor for obfuscated programming. Jarry is also prominent in Stephen Ramsay's *Reading Machinnes,* which will be presented in the section on Philosophical Programmers.

arithmetic" (195). Perl programs, on the other hand, are famous for clever and often cryptic string manipulations, highlighting one of the language's mottoes "There's more than one way to do it!" Moreover, Perl has often been combined with other languages to produce multiple codings, such that the same text forms valid source code in multiple languages, and often can be read as natural language as well.

An example of Perl poetry is given in the lexicon entry "Class Library" by Graham Harwood, a portion of which is quoted here (39).

```
# This is the core subroutine designed to give away
# cash as fast as possible
sub ReDistributeCash {
      my $RichBasterd_REFERENCE = @_;
      # go through each on the poor list
      # giving away Cash until each group
      # can afford clean drinking water
      while($RichBastard_REFERENCE ->(CASH) >= TO_MUCH) {
            foreach my $Index (keys @{Poor}) {
                  $RichBastard_REFERENCE ->{CASH}--;
                  $Poor->{$Index}->{Cash}++;
                  if( $Poor->{$Index}->{Cash}
                        => $Poor->{$Index}->{PriceOfCleanWater}) {
                        &BuildWell($Poor->{$Index}->{PlaceName});
                  }
            }
      }
}
```

The entire entry is Perl source code with large comment sections providing needed context that is imagined to be run to produce the message as an embodiment of procedural rhetoric simultaneously enacting automatic capitalist class operations. Montfort notes that "Perl poetry is a prominent modern-day form of double-coding, distinguished from obfuscated programming as a practice mainly because it is not as important in Perl poetry that the program function in an interesting way; the essential requirement is that the poem be valid Perl" (197).

Larry Wall, who invented Perl (an acronym for “Practical Extraction and Report Language”) in 1987, has called it the first postmodern computer language. Geoff Cox and Adrian Ward, authors of the “Perl” entry in the lexicon, remark that it “lies somewhere between low-level programming languages and high-level programming languages. It combines the best of both worlds in being relatively fast and unconstrained” (207). Its specification also contains eclecticisms such as giving AND higher operator precedence than OR, which Wall claims was intentional, because “modernist culture was based on 'or' rather than 'and', something he says that postmodern culture reverses” (210). Thus, Cox and Ward contend that “Wall is focusing on the eclecticism of Perl and how algorithms can be expressed in multiple ways that express the style of the programmer. . . . The suggestion is that Perl is not only useful on a practical level but that it also holds the potential to reveal some of the contradictions and antagonisms associated with the production of software” (210-211). The authors also point out the Hayles leans heavily on Perl examples in *Writing Machines* because of its emphasis on material conditions in relation to writing. She “adds the materiality of the text itself to the analysis in a similar way to those who consider code to be material. In this way, it is the materiality of writing itself that is expressed through the relationship between natural language and code—one, code, tended towards control and precision, the other, language, trending toward free form and expression” (209).

Cox and Ward also cite Angie Winterbottom's Perl poem based on Edgar Allen Poe. They note that “there are accepted techniques for reading code, and Winterbottom's poem relies on her choice of spatial arrangement and the syntactic understanding of the language itself. Only a programmer familiar with hash tables would understand that '`$blaze_of_night{moon} == black_hole`' can be read as 'The moon, a black hole in the blaze of night'” (209). The deliberate obfuscation of code for poetic and other aesthetic purposes, such as contests for the most obfuscated C program, seems to parallel Tanaka-Ishii's penchant for producing examples that are contrived for the specific topic under investigation, as well as require knowledge of programming conventions.

While there is no substitute for possessing such skills, there ought to be a difference between coming across code 'in the wild' versus these deliberately constructed examples. In the introduction to *10 PRINT*, the authors propose that “like a diary from the forgotten past, computer code is embedded with stories of a program's making, its purpose, its assumptions, and more. Every symbol within a program can help to illuminate these stories and open historical and critical lines of inquiry” (3). The book promises to examine in immense detail the famous one line program that fills the screen with a random, maze-like pattern originally written for the Commodore 64. Published in the same Software Studies series as Fuller's lexicon, Wardrip-Fruin's *Expressive Processing*, Kitchin and Dodge's *Code/Space*, and Chun's *Programmed Visions*, its distinction is that “the code with the most potential to incite critical interest from programmers, students, and scholars is that from earlier eras” (3). Studies of individual works abound in the humanities, yet Montfort et. al. feel their close study of single line of code opposes current digital humanities trends focusing on big data, “dominated by what has been variously called distant reading (Moretti 2007), cultural analytics (Manovich 2009), or culturomics (Michel et al. 2010)” (4).

While it is a stretch to compare to how a single line from a poem inspires volumes of literary and philosophical work, such as Holderlin inspiring Heidegger's critique of technology, critical focus on single line of code highlights multiple extant versions for learning, modification, and extension. “The book also considers how the program engages with the cultural imagination of the maze, provides a history of regular repetition and randomness in computing, tells the story of the BASIC programming language, and reflects on the specific design of the Commodore 64” (5). Consequently, a rhetorical aim of the book is to renew interest in learning programming via, and critical code studies of, early personal computers, “offering a hint of what the future could hold should personal computers once again invite novice programmers to RUN” (17). They argue the sensibility of studying short programs extends to studying important parts of larger systems of code, if it is going too far to suggest prolonged study of a large program like an epic poem. “*10 PRINT* shows that much can be learned about a program without knowing much of anything about its conditions of creation or intended purpose or indeed, without it even having an intended purpose” (263). It is interesting to note that early code studies occur in the domain of human-legible, print publications based because that was how programs were initially disseminated. “From the mid-1970s through the early 1980s, BASIC was known in print not only through manuals and textbooks that explicitly taught programming, but also through collections of programs that appeared in magazines and computer club newsletters. The printed materials that remain today reveal insights into the practices of BASIC users and the culture that developed among them” (182).

A point Montfort et. al. try to make by referring to the large number of magazines and books that featured programs to type in is that “the view of programs as static is even less tenable when one considers the writing, running, and distribution of programs throughout the history of computing. Custom software written for businesses has long been maintained and updated for half a century. The BASIC programs people keyed in from magazines invited users to modify them” (266). Put in the terms of Campbell-Kelly's history of software, the period of packaged media that grew out of the personal computer revolution was preceded by customized products and succeeded by continuously updated ones. Hence the authors argue “because BASIC was a hit at a unique time in the history of computing when microcomputers were becoming mainstream but before executable software distribution became widespread there may never be another language quite like it” (193). Rushkoff agrees there was a short window of opportunity in the late 1970s and early 1980s America as a golden age for learning programming, in part due to the absence of widespread executable software distribution, and also “when computers were supposedly harder to use, there was no difference between operating a computer and programming one” (139).

The authors of *10 PRINT CHR$(205.5+RND(1)); : GOTO 10* have many ambitious paths they wish to explore, reflecting no doubt its collaborative nature, but their first task is the most important, going through this one line program one token at a time, which has to occur as a fundamental act of code studies. I compare it to the necessary stepwise explication that may be given to students in a formal setting, as well as the extreme, close, hermeneutic reading done by philologists and philosophers like Heidegger's readings of Presocratic Greek fragments or Holderlin's poetry. First, the line number `10`. “The interactive editing abilities that were based on line numbers were well represented even in very early versions of BASIC, including the first version of the BASIC that ran on the Dartmouth Time-Sharing System. Line numbers thus represent not just an organizational scheme, but also an interactive affordance developed in a particular context” (10). While a BASIC program may start with any line number, `10` is frequently used rather than `1` because it encourages leaving empty lines between statements to simplify additions without the tedious task of renumbering, as early personal computers did not provide sophisticated editors like modern integrated development environments to perform renumbering on the fly. Next, the use of spaces and canonical keywords facilitates human reading and modification, acknowledging that code is more than fodder for machine translation. Comparisons could be made to the conventions of very early manuscripts before spacing and punctuation were added.

The keyword `PRINT` is a skeumorph of early scrolling paper print output before interactive editing on a video display became prevalent. The token `CHR$` is pronounced “character string,” and accesses a bank of preconfigured graphic images that are applied to the display like typographic characters. “Character graphics exist as special tiles that are more graphical than typographical, more like elements of a mosaic than like pieces of type to be composed on a press. . . . But these special tiles do exist in a typographical framework: a textual system, built on top of a bitmapped graphic display, is reused for graphical purposes” (12). This command will add exactly one of these mosaic tiles at the current 'cursor' position on the screen, which is drawn left to right, top row to bottom row. There are 255 of these tiles in the Commodore 64 ROM, and the equation in the parentheses is designed to determine which one to select. It will be either `205`, a back-slash character `\` or `206`, a forward-slash character `/`. How that happens depends on the result of a floating point arithmetic operation that is automatically truncated to its integer base by the `CHR$` operation. Commodore BASIC does all its math in floating point numbers, whereas other languages' fundamental numeric data structure is integer. The strategy here is to add a random, floating point decimal number between 0 and 1 to `205.5`, so that the result will be either `205` or `206`, depending on whether the random number is greater than or equal to `0.5`.

The authors point out the obvious use of the plus sign `+` for arithmetic, which is an abbreviation of “AND” along with the ampersand, but note that other typographical symbols were borrowed from their textual uses for multiplication and division in the Commodore 64. “Given the computer's development as a machine for the manipulation of numbers, it is curious that typographical symbols have to be borrowed from their textual uses (* indicates a footnote, / a line break or a juxtaposition of terms) and pressed into service as mathematical symbols” (13). `205.5` is being added to the output of `RND(1)`, which is a built in function producing an illusion of randomness that is actually programmed. “When RND is given any positive value (such as this 1) as an argument, it produces a number using the current seed. This means that when RND(1) is invoked immediately after startup, or before any other invocation of RND, it will always produce the same result: 0.185564016” (14). An entire chapter is devoted to randomness later in the book.

The semicolon `;` following the `PRINT` statement leaves the cursor at the next position following the string that is printed, rather than adding a new line or intervening space. The authors note the semicolon was added to version 2 of Dartmouth BASIC for output formatting. Thus its presence here “is enough to show that not only short computer programs like this one, but also the languages in which they are written, change over time” (15). Thus we can argue against Ong's judgment that computer languages are cast once and for all ahead of time, rather than emerging like natural languages. Another change to the Dartmouth BASIC specification was added by Microsoft: the colon `:` "separates two BASIC statements that could have been placed on different lines. In a program like this on the original Dartmouth version of BASIC, each statement would have to be on its own line, since, to keep programs clear and uncluttered, only a single statement per line is allowed. The colon was introduced by Microsoft, the leading developer of microcomputer BASIC interpreters, as one of several moves to allow more code to be packed onto home computers" (15). Thus, some of the clarity instrumented by adding spaces between tokens was rescinded by the later desire to pack more content onto each line of code.

`GOTO` was also not original to BASIC, but is strongly associated with it; the famously discussed denunciation by Edsger Dijkstra is often credited for prompting the move to structured high-level languages. `GOTO 10` turns the sequential printing of a single forward or back slash character into an endless loop that scrolls across the screen left to right, then fills the next line, and so on, producing the pleasing but otherwise useless maze like display from a single line of code. Almost. The authors remind us that `RUN` is an essential token though not part of the program, "what is needed to actualize the program" (16). In connecting the machine to its environment where it is used by the same agency in which the program is entered into it, it is not quite part of the stored program specification. Compare it to the exhortation to the reader that used to begin literary texts, or even the invocation to the Muses starting up the *Iliad*.

The authors claim that the Commodore 64 is the best selling single model computer to date, with over 12 million units sold (212). It lives on in emulations, which have been likened to versions of literary works, and the faithfulness of material and platform specificity can be an evaluation parameter of such emulations. "When developers produce a program, such as the free software emulator VICE, that operates like a Commodore 64, it can be considered as a software edition of the Commodore 64. . . . Instead of dismissing the emulator as useless because it isn't the original hardware, it makes more sense to consider how it works and what it affords, to look at what sort of edition it is" (21).

Like other personal computers, it shipped with a low level interface between the hardware and the rest of the system called the KERNAL. "It is the brainstem of the machine, its core, its always present, unyielding, and unchangeable center. . . . The KERNAL is intended to make machine language coding easier, providing a stable set of instructions and registers a programmer can address. Yet as enabling as the KERNAL may be, it is also structuring and limiting, the basis of the Commodore 64" (232-233). The BASIC language built into the Commodore 64 (a feature going back to the Commodore PET, along with PETSCII, its graphical character set that included "the four suits of cards, shaded patterns, and various brackets and lines" (221)) itself depends on the KERNAL to operate, a fact that complicates the notion that stored program computers are universal Turing machines that can implement the same program in different languages and platform? "The very idea of creating a program like 10 PRINT depends on aspects of the platform and the platform's facility for such a program—the presence of BASIC and RND in ROM, the existence of PETSCII, the cultural context of shared programming techniques, and of course the ability to program the computer in the first place, something owners of an Atari 2600 did not truly have" (207-208). This telling statement of why platform matters foregrounds the materiality of code and coding practices, and the differential potential of millions of Commodore 64s versus Atari 2600 VCS units to spread computer literacy versus direct manipulation.

Procedural Rhetorics of Diachrony in Synchrony

My theoretical framework is almost complete. I began with a very broad basis in critical theory, textuality studies, and media studies, layering in the social construction of technology. This set was seen reflected in a number of histories of computing, software, and networking. Next psycho-social studies of computer programmers were introduced to balance outsider perspectives with accounts from within the profession. Then philosophers of computing technologies deeply influenced by science and technology studies and Continental philosophy were introduced to respond to many of the issues raised by the field. These broad philosophical investigations were followed by original contributions from the nascent disciplines of software studies and the more focused code studies. The final, bottom layer was hinted at by the *10 PRINT* authors as they explored the various machines on which their famous one-liner has been implemented. Ian Bogost, a long time member of this group of software and code studies authors, collaborated with Nick Montfort to write *Racing the Beam: The Atari Video Computer System* as an exploration of the platform level founding the materiality—in the sense of specificity of implementation, as well as influencing the trajectories of forking paths these technologies may take by imposing structural and conventional limitations and affordances—upon which code and software goes through its life cycle of creation, use, and obsolescence, and is the site of nearly all other media production. His subsequent book *Alien Phenomenology* formally inaugurates the discipline: “in platform studies, we shift that focus more intensely toward hardware and software as actors” (100).

Platform studies appeal in a more general fashion to the point made by Kittler that the problem in the era beyond literacy is that writing is hidden in computer memory cells that are able to read and write autonomously. “The bulk of written texts including the paper I am actually reading to you no longer exist in perceivable time and space, but in a computer memory's transistor cells. . . . our writing scene may well be defined by a self-similarity of letters over some six orders of decimal magnitude. . . . It also seems to hide the very act of writing. . . . The crazy kind of software engineering that was writing suffered from an incurable *confusion between use and mention*. . . . manmade writing passes instead through microscopically written inscriptions, which, in contrast to all historical writing tools, are able to read and write by themselves” (“There Is No Software” 147). Kittler proposes the evocative image of the enormous, manually blueprinted microprocessor circuit laid out by Intel engineers as the last historical act of writing, which now becomes Castells' real virtuality, while “the hardware complexity of microprocessors simply discards such manual design techniques, instead of filling countless meters of blueprint paper, have recourse to Computer Aided Design, that is, to the geometrical or autorouting powers of actual generation” (148). As Manovich titles his most recent work, *software takes command*. This is a theme I have repeated throughout the dissertation.

To engage this writing that is partially automated, Hayles urges humanities scholars adopt new critical practices called for by electracy, such as Matthew Kirschenbaum's digital forensics and Espen Aarseth's ergodic reading. She suggests that such subdivisions of forensic and formal materiality cross in the articulation of technological concretizations (*Electronic Literature* 23). Likewise Berry describes the domain as *double mediation*, calling for reading tools like oscilloscopes and *tcpdump* to develop narratives of cyberspace constitution. He, too, notes “Kirschenbaum (2004) offers an exemplary example of researching code as a container, where he undertakes a 'grammatology of the hard drive' through looking at mechanisms of extreme inscription of magnetic polarity on the hard disk platters to understand this form of 'electric writing'” (50).

According to Kirschenbaum, extreme inscription is a limit case. “The hard drive and magnetic media more generally are mechanisms of extreme inscription—that is, they offer a practical limit case for how the inscriptive act can be imagined and executed. . . Here we will follow the bits all the way down to the metal” (92). New media scholars have ignored the hard drive despite its consistent presence in this history of electronic computing; Heim and others refer to its black box aspect, focusing instead on what Montfort calls screen essentialism, “where the vast preponderance of critical attention has been focused on what happens on the windowed panes of the looking glass. Extend electronic textuality beyond flickering signifiers on the screen” (95). Indeed, his research finds random access disk storage has been around for decades, making its public debut at the 1958 World's Fair in Brussels, where “visitors would have beheld Professor RAMAC, a four-ton IBM machine capable of offering up responses to users' queries on a two thousand year historical span ranging from the birth of Christ to the launching of Sputnik 1” (96). Using words echoing both Kittler and Chun, he describes the encounter with the first digital librarian as “like the telegraph's automatic writing or the call of the telephone, the book that can be read without being opened offers up a whiff of the uncanny, the hint of haunted media” (99).

Importantly, due to the signal processor technology at the core of the random access hard drive, we need to consider the digital to analog to digital processing that writing and reading takes on, which, per Kittler, Brosl Hasslacher calls discretization. By carefully examining the technical details, Kirschenbaum proposes grammatological primitives derived from a kind of media specific analysis: signal processor, differential, volumetric, rationalized, motion-dependent, planographic, and non-volatile but variable. Differential means signification dependent upon instantaneous changes in value rather than substance of the signal. Volumetric means traces only detectable by machinery, that cannot be read by humans.[103] Rationalized means no writing prior to formatting; contra paper, disks are a striated rather than smooth surface. Motion-dependent means inscription and reading only occur at particular rotational speed, the platter riding on an air cushion. Planographic means the surface must be absolutely smooth, and non-volatile but variable because “the magnetic substrate of a drive is one of the stickiest and most persistent surfaces for inscription we've ever devised” (105). Inscription and memory go from scarcity to superfluity, making possible Big Data as a new field of study and requiring new, distant reading techniques. “This is a sea-change in the production and recording of human knowledge, one whose implications go far beyond the hard disk drive as a technology of writing and inscription alone. As Derrida notes in *Archive Fever*, what is no longer archived in the same was is no longer lived in the same way” (107). He concludes that we must study storage if interested in texts as representations, which fits well with my call to study electronics and computer technology in general if interested in the philosophy of computing.

Kittler's take on extreme inscription is that we are now committed to writing 'under', as in underlings of the Microsoft Corporation, when we are obliged to use *Word* instead of pen and paper. The one-way function of programming languages deployed in intentionally obfuscating

103 Kirschenbaum calls the File Allocaation Table (FAT) technique used by formatted hard disks “the apotheosis of a rationalization and an atomization of writing space that began with the discrete pages of another random access device, the codex” (103).

applications and operating systems, whose innards are protected by the DMCA from casual inquiry, has corrupted a more primordial relationship between us humans and our writing tools. “This worm's-eye view did not always prevail. In the good old days when microprocessor pins were still big enough for simple soldering irons, even literary critics could do whatever they wished with Intel's 8086 Processor. . . . The higher and more effortless the programming languages, the more insurmountable the gap between those languages and a hardware that still continues to do all of the work” (“Protected Mode” 156-158).

Instead of reading from Foucault, Kittler discerns the contemporary politics of knowledge from technical handbooks about Intel microprocessors: “thus in multi-user systems, it is necessary that the programs and data of individual users are isolated, just as the operating system must be protected against user software. . . . From the 80286 on, Intel's processors are equipped with a Protected Mode that (in the words of the Siemens engineer) protects the operating system from the users, and through this protection, first allows the users to be deceived” (158). The deception afforded by protected mode and higher clock speeds corrupts “John von Neumann's classic architecture, which made absolutely no distinction between commands and data unnecessary in an era when all existing computers were still state-secrets disappears under four consecutively numbered privilege levels” (161). Foucault's power arguments are redoubled from the mute efficacy of this technical implementation. We ought to look at ICs to understand society as a form of technology studies; “one should attempt to abandon the usual practice of conceiving of power as a function of so-called society, and, conversely, attempt to construct sociology from the chip's architectures. For the present at least, it is a reasonable assumption to analyze the privilege levels of a microprocessor as the reality of precisely that bureaucracy that ordered its design and called for its mass application" (162).

As Kirschenbaum and Kittler examine the extreme inscription of hard disk reading and writing operations, and how modern computer technology veils the full capabilities of the hardware from user/consumers, we could also consider how Ethernet performs the apotheosis of telegraphy by probing the technical details that were only touched upon by Hafner and Lyon, Abbate, and Galloway. We are finally at the platform level. Montfort and Bogost enter this unexplored territory of the hard-core engineering level consideration of platforms informed by the history of material texts, programming and computing systems in what, on the surface, appears to be yet another history of a computing device, the Atari Video Computer System (VCS). "But studies have seldom delved into the code of these programs, and they have almost never investigated the platforms that are the basis of creative computing. Serious and in-depth consideration of circuits, chips, peripherals, and how they are integrated and used is a largely unexplored territory for both critic and creator. . . . Only the serious investigation of computing systems as specific machines can reveal the relationships between these systems and creativity, design, expression, and culture" (*Racing the Beam* 2-4).

The VCS was chosen because it exhibits what I call manageable complexity[104], pleasurable use, and collectibility. Monfort and Bogost argue this simple machine presented so many possibilities because it did a few things very well. "The very few things it could do well—

104 That is, complex but not too complex to be overwhelming. First generation personal computers and other early electronic consumer devices like pinball machines include all of the basic components of a stored program computer in a package that a hobbyist with schematics, an oscilloscope, multimeter, and soldering iron can usefully study and modify.

drawing a few movable objects on the screen one line at a time while uttering sounds using square waves and noise could be put together in a wide variety of ways to achieve surprising results" (15). The MOS 6507 processor chosen for the VCS limited its memory addressing to 8K at a time, which is unthinkable by today's standards. Its immediate control was via joysticks "that helped inspire the direct manipulation concept of computer interfaces in the 1980s" (25). We learn about the MOS 6532 RIOT/PIA chip that handled the input from the joysticks. We learn that graphics sprites, small matricies akin to the Commoodore 64 screen characters invoked by the `CHR$` function, are the building blocks of each game's display. A VCS programmer draws each frame of the television CRT display screen, literally pacing the beam (they admit 'racing' was chosen as a more exciting analogy). Instead of leveraging functionalities encapsulated in an automated external apparatus, "the VCS programmer must draw each frame of a program's display manually to the screen, synchronizing the 6507 processor instructions to the television's electron gun via the TIA [Television Interface Adapter]. . . . In these cases, the programmer must carefully cycle count processor instructions so they execute at the right time. While racing the beam is a catchier name, pacing the beam is more apt, since the program might have to be sped up or slowed down" (28).

The authors claim that VCS programs exhibit manageable complexity, such as not being compiled from a high level language; assembly is closer to the machine level, so that VCS ROM is literally a physical copy of the source code, and can be disassembled (if permitted). "When a program has been carefully disassembled and commented, as has been done with *Combat*, understanding the program becomes much more tractable" (33).[105] A main loop does all other

105 If disassembly is not permitted on account of violating copyright or the DMCA, then whether scholarly research

control operations during the vertical blanking interval of television electron beam, then executing kernel code that paces the beam drawing the screen.

From their detailed investigation of the platform level of the VCS, Montfort and Bogost distill a number of insights. Hardware collision detection built into the TIA afforded particular game types, “shooting or being shot by missiles, running into a wall, or consuming something,” (48) and created virtual spaces, such as the game *Adventure*, whose concept of movement from room to room persisted in two-dimensional, top-drawn games like *Ultima* and *The Legend of Zelda* series. With these games, traversing virtual space supplants narration as a procedural rhetoric. The sprite (movable bitmap) became a standard for home consoles, although it was an extraordinary challenge for VCS programmers. Using existing technical objects in new ways is a form of technological innovation, such as graphics registers for player avatars and castle walls. The world for *Pitfall* was consistently created by code using a pseudorandom sequence rather than storing a large image in the little ROM.

The VCS platform had a long run from 1977 through 1992.[106] Besides emulation, homebrew programmers continue to discover unknown capabilities of the VCS platform, and Bogost uses it for teaching. The authors invite us to consider programming languages as platforms, too, anticipating *10 PRINT*. “There have been many influential software platforms designed to run on different sorts of boxes. . . . Studies focused on the code of particular BASIC

based on such misuse legitimate is an ethical question for digital humanities. Platform studies would be a non-starter without expired patents and disinterest on the part of intellectual property holders of these obsolete devices.

106 In chapter four I introduce Pinball Platform Studies, and discuss the Bally/Stern architecture as a parallel to the VCS that flourished over nearly the same period.

programs are important to pursue, but studies that consider the programming language as a platform for computational expression will also be important" (148).

Subsequent volumes in the so-called Platform Studies series include a study of the Commodore Amiga (*The Future Was Here* by Jimmy Maher) and the Nintendo Wii (*Codename Revolution* by Steven E. Jones and George K. Thiruvathukal). The series is panned by Dale Leorke in "Rebranding the platform: The limitations of 'platform studies'" with the basic criticism that these subsequent books merely repeat the model presented by Monfort and Bogost, which diminishes the future prospects of the approach. He contends that the focus on platform invites unfounded hypotheses about they influence of a particular device, such as the relationship between Amiga and Linux, ignoring wider social and cultural connections. "But, like its predecessor *Codename Revolution*, it [*The Future Was Here*] doesn't offer a challenge to or expand platform studies to any significant extent; even as it offers a critical appraisal of the Amiga's history, it never moves beyond a kind of hybrid technical handbook/scholarly textbook account of the platform in much the same format that previous books in the series have offered" (265). Platforms have transitioned from neutral to ideological structures; that is, the the platform emphasis has been taken too far. "One can imagine an endless production line of books— one on the Magnavox Odyssey or Sega Dreamcast, another on Java or Microsoft DOS—that are valuable in themselves, but which don't expand on the established formula of the series" (266). Leorke prefers Apperley and Jayemane's work the on the material turn in game studies because it suggests more social and cultural connections. For instance, none of those books consider "tracing where the components of these platforms—the microchips, processors, cables, casing and so forth—are produced and the material labor that is put into them" (266).

Despite this criticism of the platform studies genre, there is a gem hidden in the final pages of *Racing the Beam*, however, that ties together all of the components of the theoretical framework I have presented so far. The “Afterword on Platform Studies” presents a five level diagram of digital media “that characterize how the analysis of digital media has been focused—each of which, by itself, connects to contexts of culture in important ways” (145). Graphically it appears as a 'swim lane' diagram with five rows for the levels bordered by a shared vertical column labeled 'culture and context' (see Figure 1, below). The levels they name are, top to bottom: reception/operation, interface, form/function, code, and platform. Each level represents a phenomenological order of magnitude or degree of specificity for analysis.

The topmost, very broad reception/operation level includes aesthetics, reader-response, psychoanalytic approaches, media effects and empirical studies of interaction and play; they name as notable theorists Sherry Turkle, Wolfgang Iser, Geoffrey and Elizabeth Loftus. “Although only those types of media that are interactive are explicitly operated, all sorts of media are received and understood. This means that insights from other fields can often be usefully adapted to digital media at this level” (145). The interface level includes human-computer interaction (HCI), humanities and literary comparative studies of user interfaces, visual, film theory, and art history approaches; notable theorists are Jay Bolter and Richard Grusin. Yet “the interface, although an interesting layer, is what sits between the core of the program and the user; it is not the core of the program itself. A chess program may have a text interface, a speech interface, or a graphical interface, but the rules of chess and the abilities of a simulated opponent are not part of the interface” (146). Below that, the form/function level includes game rules, the nature of simulation, and abilities of AI players. “It is the main concern of cybertext studies and of much of the work characterized as game studies or ludology” (146).

The code level, they contend, is younger still than the form/function level in terms of academic focus, although it is related to the field of software engineering and easily crosses over into technical subjects. “Code studies, software studies, and code aesthetics are not yet widespread, but they are becoming known concepts. . . . Even if the source code is not available, however, an analysis at this level of compiled code and of records of the development process can reveal many useful things” (147). Finally, the platform level is the abstraction beneath code that provides affordances and constraints instantiated at higher levels, as *Racing the Beam* sought to demonstrate. “If code studies are new media's analogue to software engineering and computer programming, platform studies are more similar to computing systems and computer architecture, connecting the fundamentals of digital media work to the cultures in which that work was done and in which coding, forms, interfaces, and eventual use are layered upon them” (147). The authors are careful to note that, despite their emphasis on the neglected platform level, “we did not shy away from mentioning some things about what games mean and how people play them, what interfaces particular games use, the particular ways that games function, and the with which they are implemented” (147). An appeal to thinking about all levels at once begs for some kind of synthesis, a grand unifying theory, but perhaps a syncretism is a more reasonable target for working towards a philosophy of computing.

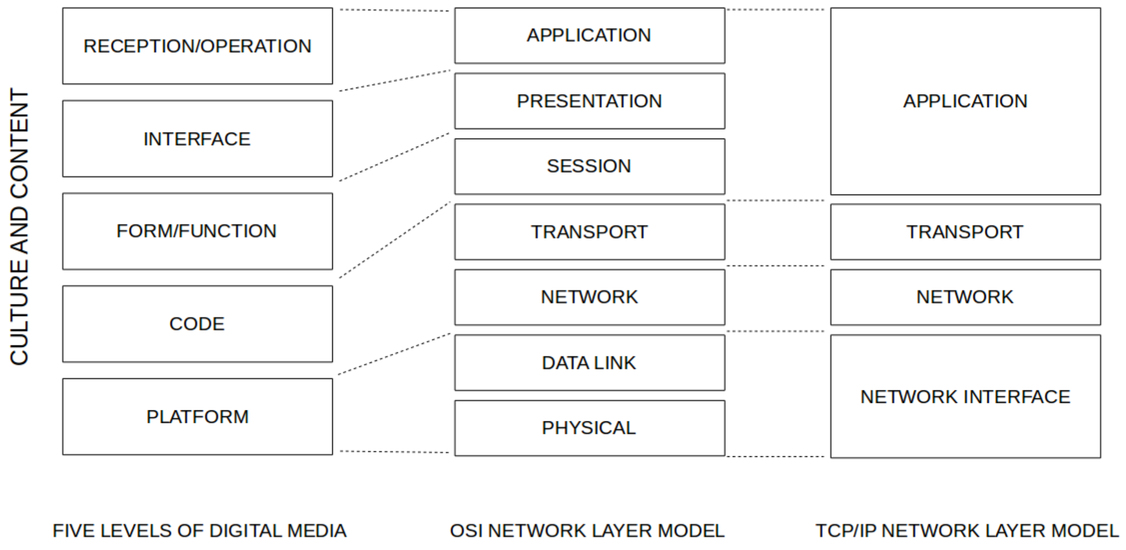


Figure 1: Comparison of Level Models

I sense an irresistible parallel between this representation of the analytical levels that digital media theorists might undertake to appreciate the culture and context of their subject matter, and the OSI and TCP/IP network layer models that are common in technical circles. Figure 1 provides a comparison of these models. Missing from Montfort and Bogost's swim lane diagram that are implied by the network model are the sundry, interrelated, immanent mechanisms that regulate the operation of each phenomenological level through time, for the diagram depicts many aspects of the same phenomena of everyday computing and digital media experience. In the network layer model this relationship is understood: each level rides atop, and is encapsulated within, the ones below it. As Galloway described protocol, power and control are immanently distributed, making mechanism and model identical. Thus, in Monfort and Bogost's diagram, the platform embodies the code level, which defines the form/function level, engaged via the interface level, so that at the top level overall aesthetic responses can manifest, particularly in the hybrid human/machine forms of reception that Hayles documents.

Granted the model is intended to be a representation of different interpretive levels, but for considering the phenomenology and operative mechanisms of everyday network phenomena occurring in massively distributed, stored program computing machinery we call cyberspace, serving up programmed visions, it is coextensive with reality. The lesson learned from studying electronics and computer technology is that, provided the epistemological transparency of free, open source components, cyberspace is comprehensible from top to bottom layer. This combined diagram, I argue, is adequate for depicting the as-is situation of the post-postmodern network dividual cyborg. As presented by Montfort and Bogost, however, the framework is lacking. Not for lack of consideration of alien phenomenology, for my analysis leans heavily on Bogost's work, and he deliberately foregrounds philosophical concepts like variable ontology and object orientations, and Montfort too is keenly interested in hardware and software as actors. Rather, this theoretical framework was briefly presented in an afterword that is primarily devoted to a justification for the authors' selecting the Atari VCS as the tutor text for articulating platform studies. To connect the diagrams requires additional conceptual components, one of which I get from Bogost and the software studies crowd, *procedural rhetoric*, and the other will be adapted from O'Gorman's new media studies, *diachrony in synchrony*.

In *Persuasive Games*, published on the back of *Unit Operations*, Bogost outlines procedural rhetoric as “the art of persuasion through rule-based representations and interactions rather than the spoken word, writing, images, or moving pictures” (ix). Others in the game studies crowd like Noah Wardrip-Fruin have identified operational logics that “are sets of standardized unit operations such as graphic logics packaged as game engine and textual logics as natural language parsers” (13). Bogost calls these procedural tropes, for they are common forms that are found in many computational media objects. Crucial to Bogost's thinking is their commensurability with forms of literary and artistic expression supporting the trope analogy. He gives the example of Socrates' trial for the ideal of efficient causation in ancient rhetoric, where “spoken words attempt to convert listeners to a particular opinion, usually one that will influence direct and immediate action, such as the fateful vote of Socrates' jury” (15).

Recall that Kittler and Chun both compare the operation of computer program code to this ideal form of rhetoric in which words do what they say. However, Bogost finds instead that digital rhetoric as conceived by many theorists “typically abstracts the computer as a consideration, focusing on the text and image content a machine might host and the communities of practice in which that content is created and used. . . . What is missing is a digital rhetoric that addresses the unique properties of computation, like procedurality, to found a new rhetorical practice” (26). He gives Manovich some credit for extending the field of traditional rhetorical forms with database logic, but finds that he fails to appreciate process intensity and favors hypertext over its supporting programmed systems. Thus it is necessary to give a name to the specific kind of persuasive power inherent in computer processes: “procedural rhetoric is a general name for the practice of authoring arguments through processes. . . . In computation, those rules are authored in code through the practice of programming” (28-29). Thus, the epitome of procedural rhetoric is working code itself, for the reliable execution of properly compiled and interpreted source code is the hallmark of stored program computing, putting aside, of course, the vicissitudes of execution. As Weinberg, Montfort et. al., and others have described the experience of reading code, once an algorithm is understood, a reader can imagine the sequence of operations effecting the result.

Videogames provide numerous examples of how “procedural rhetorics afford a new and promising way to make claims about *how things work*” (29). A game Bogost considers exemplary is *The McDonald's Videogame* by the Italian social critic collective Molleindustria. In order to play the game successfully, players must behave immorally. “*The McDonald's Videogame* mounts a procedural rhetoric about the necessity of corruption in the global fast food business, and the overwhelming temptation of greed, which leads to more corruption” (31). Other games may be provocative but not deploy procedural rhetoric, despite using animation to step through a process. They must employ representational processes to explain the actual processes they are investigating: “first, what is the relationship between procedural representation and vividness? Second, what is the relationship between procedural representation and dialectic?” (34). By vividness Bogost refers to Charles Hill's psychology of rhetorical images, in which he offers a continuum that works back from direct experience to less vivid forms like cinema down to statistics and other unemotional presentations of information. The responsive procedurality of computational media rates near lived experience, so that “procedural representation seems equally prone to the increased persuasive properties Hill attributes to vividness” (35).

Bogost uses his own concept of unit operations as substitutes for propositional content to argue that “procedural rhetorics do mount propositions: each unit operation in a procedural representation is a claim about how part of the system it represents does, should, or could function. *The McDonald's Videogame* makes claims about the business practices required to run a successful global fast-food empire. . . . These propositions are every bit as logical as verbal arguments—in fact, internal consistency is often assured in computational arguments, since microprocessors and not human agents are in charge of their consistent execution" (36).

The second aspect, dialectics, function in a broad media ecology. Bogost is careful to note that, while videogame players cannot engage in a direct argument with the game developers, or change the rules of the game play engine (for the most part, since the foss paradigm would permit such responses), they may "invite other, subsequent forms of discourse, in which interlocutors can engage, consider, and respond in turn, either via the same medium or a different one," which is the nature of dialectics (37). Thus we need to distinguish between the ability to raise procedural objections by altering game play, and the emergence of dialectical reasoning about the subject whose proceduralities are represented in the videogame. He gives the example of *The Grocery Game*, which allows modification of rules of shopping by automating otherwise too costly behaviors for saving money with coupons and timed bulk purchases at particular grocery stores, and peripherally allows criticism of game mechanics in message boards to substitute for modifying code. "While the game does not provide the user with direct access to the search algorithms that generate its lists, so that a user could wage these objections in code, it does provide a flourishing community of conversation. . . . The community discourse at the game's message boards are not always related to objections to its underlying procedural rhetoric, but the availability of this forum facilitates active reconfiguration of the game's rules and goals" (39-40).

Interactivity is a key component of procedural rhetorics because play, understood as "the possibility space created by processes themselves," which we explore by manipulating the game's controls, is the mechanism by which arguments come to be understood, rather than through discursive means. Bogost borrows the enthymeme from Aristotelian rhetoric, “the technique in which a proposition in a syllogism is omitted; the listener (in the case of oratory) is expected to fill in the missing proposition and complete the claim” (43). Procedural enthymemes complete the claim by playing the game. “A procedural model like a videogame could be seen as a system of nested enthymemes, individual procedural claims that the player literally completes through interaction” (43). Of course, we must distinguish this action from the mere persuasiveness games emit to entice us to keep playing them. “Instead, I am interested in videogames that make arguments about the way systems work in the material world. These games strive to alter or affect player opinion outside of the game, not merely to cause him to continue playing” (47). Moreover, so-called serious games designed for educational purposes may not interrogate institutions and worldviews. Nonetheless, we can connect the notion of 'serious' to 'underlying system structure' that is also foregrounded by critical code studies; “procedural representation depicts how something does, could, or should work: the way we understand a social or material practice to function” (58).[107]

Through procedural rhetoric games exercise often clever and unexpected biases in our actions, which when uncovered and critically engaged potentially inspire radical change. James Paul Gee's extensive analysis of what video games have to teach us about learning and literacy, in the book bearing that title, explains how active and critical learning reinforce the importance of situated meanings in semiotic domains over sheer informational content. Gee contends “the learner needs to learn not only how to understand and produce meanings in a particular semiotic

107 Bogost employs Badiou's terms situation, multiplicity, count-as-one, state, and event to form the basis of seriousness underlying structure of a system.

domain but, in addition, needs to learn how to think about the domain at a meta level as a complex system of interrelated parts. The learner also needs to learn how to innovate in the domain—how to produce meanings that, while recognizable to experts in the domain, are seen as somehow novel or unpredictable" (25). Critical learning seems to imply ergodic relationships to texts in general, including games.[108] Gee comes close to describing procedural rhetoric in the context of critical learning when it occurs through game play. "What we are dealing with here is talking and thinking about the (internal) design of the game, about the game as a complex system of interrelated parts meant to engage and even manipulate the player in certain ways. This is metalevel thinking, thinking about the game as a system and a designed space. Such thinking can open up critique of the game. It can also lead to novel moves and strategies, sometimes ones that the game makers never anticipated" (34-35).

Gee articulates what active and critically played games can do with their ergodic properties that print texts cannot: "*they situate meaning in a multimodal space through embodied experiences to solve problems and reflect on the intricacies of the design of imagined worlds and the design of both real and imagined social relationships and identities in the modern world*" (40-41). In doing so he sets up a tension between playing games and understanding program code that Bogost explicitly calls out while discussing Sherry Turkle's criticism of *Sim City* for not offering 'policy knobs' to adjust its deterministic relationships between taxation and social policy. "Turkle's real beef is not with *Sim City*, but with the players; they do not know how to play the game critically. Understanding the simulation at the level of code does not necessarily

108 In ergodic media, non-trivial effort is required to allow the reader to traverse the text, such as choosing hyperlinks to navigate a work of electronic literature or navigating a virtual world in the typical videogame.

solve this problem. . . . Rather than addressing this problem from the bottom up through code literacy, we need to address it from the top down through procedural literacy" (63-64). Bogost, who is a seasoned game developer as well as a media critic, cannot be ignored on this point. "Publicly documented hardware and software specifications, software development kits, and decompiled videogame ROMs all offer possible ways of studying the software itself. Such study can shed important light on the material basis for videogame experiences. An understanding of code supplements procedural interpretation. In particular, a procedural rhetorician should strive to understand the affordances of the materials from which a procedural argument is formed" (63). He seems to be rejecting the bottom up argument of Rushkoff to program or be programmed, instead reasoning that a top down paradigm is preferred.

The short history Bogost gives of procedural learning from Kay and Goldberg's Smalltalk language and Papert's Logo to the RAPUNSEL project leads to this rejection because a programming emphasis excludes the built-in procedurality of videogames experienced by merely playing them. "More broadly, I want to suggest that procedural literacy entails the ability to reconfigure concepts and rules to understand processes, not just on the computer, but in general. The high degree of procedural representation in videogames suggests them as a natural medium for procedural learning" (245). He draws an analogy between the perceived value of learning foreign languages like Latin to programming languages, with the important qualification made by Dorothy Sayers in the mid twentieth century, "rather than suggesting that the exercise of Latin, or mathematics, or history themselves strengthen the mind through generic exercise, Sayers' proposes that the embedded logics of such subjects provide the tools necessary to interrogate new, unfamiliar questions. These tools become the basis for living a productive adult life, or for interrogating a new, more advanced subject at university (the equivalent of the medieval quadrivium, which follows the trivium)" (247).

In its ideal application, "Latin would be allowed to oscillate between its formal and cultural registers; on the one hand, the language itself possesses formal features of synthetic inflection, specific cultural output can be consumed or created" (249). Yet this vision failed to materialize, and Latin remains a structured mental exercise. By extension, "computers constrain expression even more, through both hardware and design of programming language. . . . In many ways, programming and Oulipian writing offer even stronger evidence for the benefits of systematic training than Latin; after all, natural language is subject to human failing and misinterpretation" (249). At the same time, the exercises budding programmers are given to tackle amount to no more than the manufactured Latin lessons of yore. "It is precisely *specific areas of experience* that have been expunged from our understanding of constructivist learning and procedural literacy in particular; it is also the corrective for the practice of divorcing subject-specificity from learning" (250).

Echoing a theme that has been raised by many of the theorists already surveyed, procedural literacy spends too much time representing dynamic systems rather than investigating the cultures embedded in them. "Like the cultural and formal specificity of Latin versus Inuit or the formal properties of C versus LISP, the procedural affordances of a computer operating system *matter*: they constrain and enable the kinds of computational activities that are possible atop that operating system" (251). By shifting the emphasis to videogame criticism, more of the embedded social and cultural components can be teased out, whether by traditional methods or intuited via the completion of procedural enthyemes through playing them.

Because Kitchin and Dodge's code/space, and Levy's knowledge space are now immanent to the built environment, and the extended mind necessarily participates in it, not understanding their schematism of perceptibility leaves humans floating on top of these data streams as consumers enjoying flickering signifiers like the prisoners in Plato's cave, or their future counterparts as the lazy, recumbent passengers of the *Axiom*. Worse, they could be heavily confused network actors who make poorly informed decisions when sharing personal information, engaging in social media activities, or making purchases. The new ontologists who create code/space and everyday computational objects are not merely mimicking natural processes and artifacts, but fashioning new, often idiosyncratic phenomena and hence the possibility of new procedural tropes in their own right. Therefore, there is a risk in following Bogost's top down approach to procedural literacy of misinterpreting the tropes themselves through the distortion of programmed visions. That is why I have presented the side-by-side model of interpretive levels and network layers for the highly engineered environment of cyberspace. Procedural rhetoric is an engineering mindset, interpreting phenomena by understanding how it works.

To restore the position of programming as the quintessence of procedural rhetoric and complete the construction of my theoretical framework, I wish to appropriate a concept from Marcel O'Gorman's new media studies *E-Crit* that he in turn repurposed from literary criticism. The context is a meditation on the use of images in scholarly texts—always a sort of remainder, assumed to be periphery elements added to essays that may alter the readings of canonical texts but not change the processes of reading and writing substantially. He counters this stereotype with his version of Ulmer's heuretics, or logic of invention, as an alternative to the hermeneutic circle, where pictures can be extremely productive generative instruments, which he calls hypericonomy. He explains that “to understand pictures as generators is to view them much in the same way as Lecercle describes the pun and other forms of metaphor, all of which fall into the category of the remainder, which Lecercle describes as instances of 'diachrony-within-synchrony'. . . . The notion of 'diachrony-within-synchrony' points to the capacity of the remainder to interrupt our synchronic understanding of a word by invoking a diachronic association” (12).

The simplest example of diachrony-within-synchrony occurs when a personally charged word is encountered while reading a text, for instance when a generic term is also a proper name, or the sequential, punning use of similar words creates a chain of meaning into a very specific, significant image. Its singular, synchronic, contextual function is multiplied into a family of resemblances. O'Gorman presents a vise. “As a pictorial pun, the vise can be diachronically transformed into a printing press, the brackets in a set of Ramist dichotomies, and the hands of William Blake's Nurse, all of which play a seminal role in the chapter” (13). Lecerle also offers the term 'indirection' to describe an underdetermined process of interpretation in which “all the possible meanings of a metaphorical phrase are present at once, without the order they are given in a dictionary, which is normally based on historical principles and treats the 'literal' meaning as primary” (13). Perceptual stimuli are unconsciously funneled into common sense, although some schizophrenics and highly creative people seem to delay the assignment of meanings to words, images, and other perceptions.

O'Gorman argues “that it may be possible to teach creativity by providing individuals with an apparatus for knowledge-building that allows them to maintain access to a more diverse range of information. . . . I would like to believe that one purpose of hypericonomy is to provoke or mimic the fluidity of creative thought and crystallize it, transforming *délire* or schizophrenia into a theory and a discursive practice" (14). Humans equipped with dynamic media and working code could foster creativity through synaptogenesis without the psychological destabilization. That is not my purpose for appropriating this term, however. Rather, it is to deploy the idea of diachrony in synchrony as a procedural rhetoric for the phenomenology of cyberspace, inspiring the digital humanities practice I call critical programming.

Ontological and epistemological models featuring synchronic levels and layers are commonplace. They are at the core of structuralism, according to Barthes, who writes "the goal of all structuralist activity, whether reflexive or poetic, is to reconstruct an object in such a way as to manifest thereby the rules of functioning (the functions) of this object" ("Structuralist Activity" 149). He continues with the qualification that structure, in this sense, "is therefore actually a *simulacrum* of the object, but a directed, *interested* simulacrum, since the imitated object makes something appear which remained invisible, or if one prefers, unintelligible in the natural object. . . . the simulacrum is intellect added to object, and this addition has an anthropological value, in that it is man himself, his history, his situation, his freedom and the very resistance which nature offers to his mind" (149-150). In the context of semiotics, structuralism situates meaning in the holistic system, having particular implications for special cases like self-referential statements. For Tanaka-Ishii, "the generative model explains this structural aspect of the system in relation with how the signifier articulates the signified: the speculative introduction of a signifier generates a meaning consisting of an ensemble of content and uses, thus forming a structural system" (152).

In the case of technological phenomena of the sort that have been the subject of this dissertation, however, they are essentially simulacra coextensive with their structuralist interpretation. As Hayles puts it, “although it is true that digital technologies can create objects for which there is no original (think of *Shrek*, for instance), the technology itself is perfectly representable, from the alternating voltages that form the basis for the binary digits up to high-level languages such as C++” (*Electronic Literature* 180). Edwards describes nested levels from hardware electronics, digital logic, machine language, assembly language, high level languages, to user interfaces and operating systems. “The operation of computing machinery can be described at a number of levels. . . . Each level is conceptually independent of the ones below and above, while remaining *practically* dependent on the lower levels” (244). Abbate explains that “the protocol stack model would quickly come to dominate the way people thought about organizing networks precisely because it offered a blueprint for reducing the complexity of network components while increasing the predictability of the system as a whole” (53). A layered system has technical and social implications; “organized as a set of discrete functions that interact according to specified rules. . . each higher-level function builds on the capabilities provided by the layers below. . . . It makes the technical complexity of the system more manageable, and it allows the system to be designed and built in a decentralized way” (51).

In the synchrony of instantaneous network operations, multiple levels perform their unit operations in their own diachronic processes. For Galloway protocol is doubly materialist as networked bodies and as conditions of experience. “From the perspective of protocol, there are no biologies, no technologies, only the possible interactions between vital forms which often take on a regulatory, managerial, and normative shape. . . . Layering is a central concept of the regulation of information transfer in the Internet protocols” (xx-xxi). We can compare his analysis of executable metalayers encapsulating code, making them hyperlinguistic rather than sublinguistic, to their materiality as articulated by Berry, who writes “the hacker's close relationship to code displays the power of protocol, particularly its ability to compel autonomous actors toward a more vital or affective state within their particular distributed milieu” (167). Inaccuracies exist when the models are imperfect, neglecting the vicissitudes of execution, for instance, or impenetrable black boxes on account of obfuscation by machine operations such as compiler optimizations (Chun) and protected modes (Kittler), or intellectual property restrictions enacted by the code itself (Lessig). Moreover, because digital technologies are able to exercise agency, Hayles proposes that “the layered architectures of computer technologies enable active interventions that perform actions beyond what their human programmers envisioned” (181). Failure to engage the various layers relegates our comprehension to programmed visions.

Methodology

Returning once more to the image of Foucault's dust, “the dust of events, actions, behavior, opinions – 'everything that happens',” suggesting an epistemology that engages the world when it was *not* infiltrated by code/space, and the documentary work manually done by police powers was *not* autonomously performed by machine algorithms searching capta trails of every move ever made in cyberspace, and through triangulation, the geographical coordinates of bodies, I hope I have made the argument that old tools need to be put aside to solve new problems—philosophy included. I have attempted to document a convergence of theories arising from diverse disciplines towards a dynamic, layered model. Recalling, on the one hand, Iser's point that in the humanities theories divide between those that “have to be transformed into a method in order to function, and those that are applied directly, retroactively undergoing a diffraction of their categories,” the distinction between theory and method is somewhat arbitrary. Indeed, the nascent disciplines of software studies and code studies exhibit the immaturity of theory in progress consonant with McGann's contention that poiesis-as-theory may be the most appropriate approach for digital humanities. On the other hand, the SCOT approach has been producing many very well researched texts that can contribute to the formation of a canon for the history and philosophy of computing.

The methodology for this dissertation ought to require immersion in both pools, but time and space constraints force the latter to be only briefly considered and reserved for future scholarship under the moniker Philosophical Programmers. A quick pass through the outline of this project prepares for the research methodology I am explicitly promoting with this dissertation that flips this around to Programming Philosophers. Critical code studies engage phenomenological and hermeneutic approaches to philosophical study of code, architecture, and documentation; at the critical programming intensity, I insist practitioners develop liftetime portfolios of and by working code, that is, engage in system integration projects they use in their everyday and scholarly practices.

Philosophical Programmers

What should be our critical equipment for the current period? Requoting Latour from the first chapter, “I simply want to do what every good military officer, at regular periods, would do: retest the linkages between the new threats he or she has to face and the equipment and training he or she should have in order to meet them and, if necessary, to revise from scratch the whole paraphernalia. This does not mean for us any more than it does for the officer that we were wrong, but simply that history changes quickly and that there is no greater intellectual crime than to address with the equipment of an older period the challenges of the present one" (231). Many of the philosophical bases that have been established as the antecedents for pondering the current technological era—Nietzsche, Heidegger, Horkheimer, Adorno, Lacan, Barthes, Derrida, Foucualt, Deleuze, Guattari—are ill-equipped to handle electronic circuits, system integrations, language specifications, compiler designs, protocol declarations, and so on. "The question was never to get away from facts but closer to them, not fighting empiricism but, on the contrary, renewing empiricism," Latour assures us (231). If we are going to get close to the machine, then we need to look for a new set of foundational thinkers to complement if not supplant much of the existing paradigm.

There is a pending research agenda that can be commenced following the SCOT principles directed to the little explored discourse networks of the *philosophical programmers* who are credited with developing the machinery, languages, network protocols, operating systems, and applications of the post literacy epoch. While not professing to be philosophers, they have influenced the development of technology systems in the United States, giving them distinct characteristics that now reflect back on societies that have grown up using them. In this section I will sketch a future research agenda that gives voice to the technologists who are pioneers of babelization, distributed network visionaries, and new ontologists. Many of these individuals are referenced by contemporary historians and theorists whose work builds my theoretical framework. It will examine the work of early pioneers Bush, Turing, Burks, Goldstine, von Neumann, Licklider, Simon, Kemeny, Kernighan, Ritchie, in addition to programming languages of the period reviewed by Knuth and Pardo. This group is followed by developers of network protocols and operating systems, including Engelbart, Shneiderman, Stallman, Gates, Torvalds. Smith, Stroustrop, and others become what I call the new ontologists, whose excursions into metaphysics for the sake of defining software objects and programming styles have become the basis of metaphors in everyday conversation.

Edwards proposes that “by exploring how such subcultures use the COMPUTER metaphor in articulating key cultural formations such as gender and science, we can discern some of the salient patterns in cyborg subjectivity,” noting the territory already staked out where “Turkle thus establishes a dualism at the heart of cyborg subjectivity, a hard self and a soft one. This is an evocative, problematic, and paradoxical dichotomy” (166-167). The second part of this future research agenda will examine individual application developers, exploding the binary division of programming styles Turkle identified as hard mastery and bricolage to propose many other styles and habits, drawing from anthologies and interviews by Lammers, Shasha, Lazere, Hafner, Lyon, Oram, Wilson and others in addition to writings of Weizenbaum, Winograd, Flores, Morningstar, Farmer, and Ullman. It concludes with a look at ethnographies and auto-ethnographies of coding places done by Brooks, Rosenberg, MacKenzie, Ensmenger, and Takhteyev, where working code and reflection on the communities producing it are consciously combined. But this must wait for future endeavors.

Working Code Places

Felix Guattari's intriguing essay “Machinic Heterogeneis” foregrounds vitalist conceptions of machines, arguing “while mechanistic conceptions of the machine rob it of anything that can differentiate it from a simple construction *partes extra partes*, vitalist conceptions assimilate it to living beings, unless the living beings are assimilated to the machine” (13). His example of Legba fetish reinforces Latour's claim that we know less about our local technological milieu than tiny populations of alien, archaic societies, which nonetheless seem better equipped than hegemonic subjectivities to grasp multivalence of alterity. Latour asks rhetorically in *We Have Never Been Modern*, “paradoxically, we know more about the Achuar, the Arapesh or the Alladians than we know about ourselves. . . . Is anthropology forever condemned to be reduced to territories, unable to follow networks?” (116). In “Why Has Critique Run Out of Steam?” he makes the charge that academia has been slow to prepare for new threats, tasks, and I would add, new tools.

A good example is Derrida's comment in *Archive Fever*, which goes well with the prior discussion of extreme inscription by Kirschenbaum. “Without waiting, I have spoken to you of my computer, of the little portable Macintosh on which I have begun to write. . . . I asked myself what is the moment proper to the archive, if there is such a thing, the instant of archivization strictly speaking, which is not, and I will come back to this, so-called live or spontaneous memory (mneme or anamnesis), but rather a certain hypomnesic and prosthetic experience of the technical substrate” (25). Then there is Heidegger, quoted by Michael Heim in *Electric Language*, fearing that “the language machine regulates and adjusts in advance the mode of our possible usage of language through mechanical energies and functions. The language machine is

– and above all, is still becoming – one manner in which modern technology controls the mode and the world of language as such” (81). Such premonitions of word processing have launched many rebuttals that inevitably turn away from the machinic, try to put it into perspective, and so on. Yet as Thomson's article cited in the first chapter argues, Heidegger sought the promise in the midst of technology's greatest danger: “Midnight, seen otherwise, is dawn. . . . Heidegger believes that we discover what saves us precisely by deeply experiencing what most endangers us, and he first tries publicly to communicate his way of making sense of this idea in terms of 'the promise'" (157-158). The methodology I propose aims directly into the danger, suggesting we make a study of *working code places*, to which I assign the identifier PHI to acknowledge the unsuitability of print to represent their machinic vitalism.

Burks, Goldstine, and von Neumann introduce us to PHI. “In a special-purpose machine these instructions are an integral part of the device and constitute a part of its design structure. For an all-purpose machine it must be possible to instruct the device to carry out any computation that can be formulated in numerical terms. Hence there must be some organ capable of storing these program orders. There must, moreover, be a unit which can understand these instructions and order their execution” (1). It is not an exciting intelligence like human consciousness, nascent for so long, latent in designs and physical structures, only coming to life with the emergence of higher address bit-width systems. The genius of the stored-program computer is that “if, however, the orders to the machine are reduced to a numerical code and if the machine can in some fashion distinguish a number from an order, the memory organ can be used to store both numbers and orders” (1). Storage of numbers and orders in the same memory device implies a variable ontology of the sort Harman and Bogost describe, which is at best simulated in written texts with the aid of human interpreters who can appreciate literary conventions, puns, and so on. We do not immediately have an analogy beyond this poor comparison with written texts; even the hyperlink is inadequate now that unit operations connected to PHI can bring to the fore any element spanning an entire corpus that formerly consumed whole careers of academics to traverse, and can in turn be routinely processed one after another, as Manovich has lately made his focus drawing on findings from Big Data analytics.

The first point I want to make is that the study of texts and technology leads to working code places by carrying performativity from the exclusive domain of natural automata into one shared with the artificial. Before we began writing with the intent of steering mechanical souls with words—programming as rhetoric—the aim of communication was always other humans. This leads to very cumbersome examples when trying to make the sorts of philosophical points Derrida attempts in *Glas* and *Postcard*, which Neel parenthetically remarks “do not represent the models for a new kind of writing that can now appear in the wake of Derrida's deconstructive apocalypse. For an argument that they *do* represent such a new writing, see Ulmer” (104).[109] Neel asserts, instead, that “four effects of Derridean analysis seem obvious: self-contained, complete meaning is impossible; expectations of reading change; conceptions of writing change; and, as a result of these first three, a new sort of apocalypse is at hand, though what sort of appearance it may make remains impossible to predict” (104). In this context, Neel's conclusion that “our

109 Barthes, on the contrary, succeeds repeatedly with his brief, humorous vignettes that rely on procedural enthymemes in which the target of the narrative finds its meaning by churning away in the contextual setting Barthes assumes his reader also encounters it, such as Operation Margarine. He explains a part of the structure and mechanism by which it operates, and then presents an everyday example for the reader to fill in the details.

resistance to Derrida's readings lets us know how the student writer feels about the way we treat student texts. . . . The teacher's incisions, like Derrida's, depend on clipping out examples of the writing, dismembering the text in order to expose its operations” (134-136).

This mode of reading and writing resembles interactive debugging: stepping through source and object code simultaneously, watching the machine operations that have been translated by the compiler from the groping human attempt to complete a procedural enthymeme. In this sense *working* code is an activity more than a particular artifact worth collecting. The other sense of code *that works* distinguishes it from what Hayles calls Creole discourse. “Compounded of language and code, it forms the medium through which the origin of subjectivity can be re-described as coextensive with technology. Just as these hybrid articulations do not exist apart from their penetration by code, so the subject does not exist apart from the technology that produces the creole describing/creating the techno-subject” (*Writing Machines* 53). For example, I have found myself mixing technical acronyms, Backus-Naur Form (BNF), pseudocode as well as using actual program code in my notes to illustrate some point or convey an idea. Artists like Mez use this sort of Creole in their everyday Facebook posts.

According to Hayles, code work ranges from machine readable and executable to broken code. “Code work in its purest form is machine readable and executable, such as Perl poems that literally have two addressees, human and intelligent machines. More typical are creoles using broken code, code that cannot actually be executed but that uses programming punctuations and expressions to evoke connotations appropriate to the linguistic signifiers” (*Electronic Literature* 20-21). Broken code and creole forms abound in the work of lan Sondheim, MEZ (Mary Ann Breeze), and Talan Memmott. As Chun describes it, “this notion of source code as readable as creating some outcome regardless of its machinic execution underlies codework and other creative projects. . . . Regardless of whether or not it can execute, code can be, must be, worked into something meaningful. Source code, in other words, may be sources of things other than the machine execution it is supposed to engender" ("On Sourcery" 312). However, I want to set broken code aside and stay focused on human-readable text intended for conversion into machine instructions. Hayles expresses "the primary difference is the fact that an electronic text is generated through multiple layers of code processed by an intelligent machine before a human reader decodes the information. McGann argues that print texts are also coded, but this point relies on slippage between the narrow sense of code as it applies to computers and a more general sense of code as Roland Barthes envisions it, including codes of social decorum, polite conversation, and so on" (*My Mother was a Computer* 108).

This preprocessing of multiple layers implied by diachrony in synchrony models aligns with her description of the emergent materiality within a broad definition of text that includes "verbal, visual, oral and numeric data, in the form of maps, prints, and music, of archives of recorded sound, of films, videos, and any computer-stored information" as embodied entities on account of "the interaction of its physical characteristics with its signifying strategies. . . . Because materiality in this view is bound up with the text's content, it cannot be specified in advance, as if it existed independent of content. Rather, it is an emergent property" *(*103). The primary emergent materiality of working code is its performance, execution by the machinery, which may in turn enable further agency by the machines or combined machine/human assemblage. Hayles offers the term Work as Assemblage (WaA), "a cluster of related texts that quote, comment upon, amplify, and otherwise intermediate one another," as a parallel to Deleuze and Guattari's Body without Organs (BwO) to describe the programmed computer as author, in contrast to the familiar division between print text and human reader as locus of decoding agency (105-107). Thus, when speaking about electronic literature arising from the working code of theorist-practitioners, “rather than being bound into the straitjacket of a work possessing an immaterial essence that textual criticism strives to identify and stabilize, the WaA derives its energy from its ability to mutate and transform as it grows and shrinks, converges and disperses according to the desires of the loosely formed collectives that create it” (107).

A second characteristic of working code places beyond this intended, ongoing performativity of its assemblages, in contrast to the aesthetic and critical appreciation of a static work product, is the feedback from the programmer. Hayles has already defined electronic texts as processes rather than objects, and the way humans experience electronic texts ranges far broader than traditional reading settings, for example the incomprehensible temporal orders foregrounded in Poundstone's *Project for Tachistoscope*, where “as the kinds and amounts of sensory inputs proliferate, the effect for verbally oriented users is to induce anxiety about being able to follow the narrative while also straining to put together all the other discordant signifiers” (*Electronic Literature* 140). She relates this experience to Antonio Damasio's research in bodily knowledge focusing on the role of the limbic system and viscera in traumatic events. “In this context, literature can be understood as a semiotic technology designed to create or more precisely, activate feedback loops that dynamically and recursively unite feelings with ratiocination, body with mind” (134).

Natural language intersects code in comment lines and the underlying syntax of programming languages, and thus working code acts “like linguistic levers, giving a single keystroke the power to change the entire appearance of a textual image" ("Print is Flat, Code is Deep" 81). At the same time, the levers are microscopic and incredibly fast. Hayles describes the transliteral morphing explorations of algorithms underlying phonemic and morphemic relations by John Cayley, "a computational procedure that algorithmically morphs, letter by letter, from a source text to a target text. . . . The complexity of these relationships as they evolve in time and as they are mediated by the computer code, Cayley conjectures, are the microstructures that underlie, support, and illuminate the high-level conceptual and linguistic similarities between related texts" (*Electronic Literature* 145-147).

My point is simple and obvious: if "Cayley suggests that if a user watches these long enough while also taking in the transliteral morphs, she will gain an intuitive understanding of the algorithm, much as a computer game player intuitively learns to recognize how the game algorithm is structured" (147), then the experience of working code over long periods ought to instill intuitive understanding of machinic cognition, even if the actual operations are beyond our comprehension due to their high frequency, microscopic unit operations.

This intuitive grasp of what I call the alien temporalities of machine embodiment is not only coincidentally, as in the case of Cayley's experiments, but necessarily guided by technical know-how, which emerges through long habituation to working code places. This is the position Hayles reaches in her recent book *How We Think*. "Grasping the complex ways in which the time scales of human cognition interact with those of intelligent machines requires a theoretical framework in which objects are seen not as static entities that, once created, remain the same throughout time but rather are understood as constantly changing assemblages in which inequalities and inefficiencies in their operations drive them toward breakdown, disruption, innovation, and change. Objects in this view are more like technical individuals enmeshed in networks of social, economic, and technological relations, some of which are human, some nonhuman” (13).

Working code places imply deliberate attention to the materiality of their production and deployment. If ancient rhetoric is the steering of souls with words, working code is cybernetic, influencing both machine and human action. Knuth's proposal that we analyze every process that our computer executes in one second requires a radical repositioning of this imaginary observer. Yet the tendency among humanities theorists is to argue from the upper positions of Montfort and Bogost's diagram, in the registers of reception, interface, and software studies. Bruce Janz, in “The Betweenness of Code,” suggest a number of ways people think about code in relation to place and space: as the ultimate threat to place and the sanctity of authentic handiwork, as an enabler of free action in space lauded by techno-evangelists, or as deeply entwined with materiality, as Berry argues (np).

However, Berry's phenomenology is ultimately tied to Deleuze, and Deleuze seems to put a negative spin on technology that affects the acceptance of programming as a valid critical scholarly research methodology, an unfortunate side effect of his popularity in philosophy of technology, and now philosophy of computing studies—at least those arising from outside of computer science. Moreover, philosophical projects that feature mapping disciplinary territories —which is what Janz intends by 'space' versus 'place'—often yield a false sense of understanding the subject matter. “While the map may give the feeling that understanding has been achieved, it is not necessarily so. Like Foucault's description of animals in the Chinese encyclopedia, there is always another way of sorting out the world, and the fact that the new way may be unfamiliar does not in itself mean that it is wrong" (*Philosophy in an African Place* 84). At the risk of insulting the putative philosophers of computing who are not themselves lifelong programmers, I contend that too many treatises have been written that lump Basic and C++ together as belonging to the Emperor (Microsoft), FORTRAN and COBOL as embalmed, and many that look like files from a long way off (HTML and XML).

As Janz puts it, "most philosophers assume that philosophy is a universal enterprise, and geographical designations simply point out historical contingencies, not essential differences" (83). He suggests that we *listen* to language, because "analysis of its structure alone will not yield a self-reflective philosophy" (156). Thus "the significance may lie not in depth, but in breadth. It may be that philosophy in an African language is a worthwhile thing to pursue precisely because it is not English, French or German. It exists at a distance, and that distance itself forces new ways of understanding something" (160). Substitute "African language" with "programming language." As I hinted at the outset of this section quoting Latour, we seem to know more about "the Achuar, the Arapesh or the Alladians than we know about ourselves." If the objection is raised that source code repositories are not the place to record grand insights, Janz reminds us that philosophy does not inhere in its artifacts, which are instead traces of philosophy occurring; as thought questioning itself, it is a present concern (178).

To the extent that contemporary philosophers have flourished through their interactions with technical disciplines, such as those mentioned throughout this dissertation, a further justification for exploring territories close to the machine arises: payback. Following Derrida, Janz declares "philosophy owes a debt to the society in which it finds itself. It is not only materially sustained by the institutions of these societies, it also finds its material for reflection in these communities. . . . There is at least the debt, then, that philosophy must speak back to the place that gave it voice” (205-206). In case of computing to speak back to places that gave philosophy voice suggests reentering 1980s personal computer culture to better understand the current Internet age, or even less accessible platforms that preceded them.

In articulating his method of platial analysis, Janz credits first Heidegger and more importantly Merleau-Ponty for suggesting and then foregrounding the importance of place for making different types of knowledge possible. "One might take his notion of embodied knowledge as requiring a sense of place for fulfillment. The two together, along with any mediating devices (such as technology) that make the connection between body and place possible, we will call the milieu” (8). The milieu of working code necessarily involves mediating devices that not only convert microscopic extreme inscriptions into viewable representations, but also the programmer's workspace, the platform consisting of operating system, language compiler, additional code libraries, object code linker, run time manager, and so on that constitute the performance of working code. “When we ask about place, therefore, we ask about the type of knowledge that is made possible in a particular milieu. To a certain extent, the knowledge itself will be a function of the milieu, and both the place and the body that knows the place will find their identity in the kind of relationships possible in the milieu” (8).

Cast in this perspective, working code places radically transcend the depiction of source on a printed page, and even screen shots or videos of run time performance because the milieu exceeds these representational forms and is instead coextensive with the built environment, what Nigel Thrift calls the technological unconscious. To even begin to adequately purview working code places is to appreciate the diachrony in synchrony of variable ontology network layers in which the machinic and human are intermingled. Thus when Janz proposes the 'topeme' as the smallest intelligible unit of place, it must not be thought in the reductionist terms of space—taken down to the smallest discernible physical units, for that is like speaking about computing as bits, positive and negative voltages. Instead, he refers to De Certeau's space as practiced place, suffused with the meaning of practices, “or place that has had the meaning of practices imposed upon it” (22). Janz further develops this notion regarding the betweenness of code, likewise differentiating digital place from space. “Code is the basis of digital place. Digital place is assembled, emergent place. It resists elements of the phenomenological sense of place—it is scene to phenomenology's dwelling” (40).

The procedural enthymeme I add to this analysis is reached when we try thinking with respect to spaces and places where one worked and works code, such that even simulacral, virtual realities emanate practices. An insight the digital humanities gain asserting a continuum between engineering knowledge and digital place (scene) is working code as place, the intrinsic value of spending time writing software and tinkering with electronic machinery.

Recalling the initial hostility toward programming languages voice by Ong, and the suspicion toward computer technology latent in Heidegger, Deleuze, even Derrida, and the blatant accusations made by Winner, Weizenbaum, and others concerning the negative impact of computationalism for the majority of humans, it is fitting to compare the attitude toward working code places and African philosophy as Janz situates his work. “Africa is a good starting point for this study precisely because of the history of its dismissal. . . . African philosophy has tended to focus on subject matter outside of itself, and not seen its own work as supporting philosophical reflection” (25). He contends that too much effort has been spent justifying or defending the legitimacy of African philosophy in the first place, in part because the canonical concepts circulating in the discourse networks of African philosophy “have been used spatially instead of platially, that is, they have been used to establish and/or defend a territory known as African Philosophy rather than generate new concepts within African philosophy” (28).

Ong likewise sets up a spatial analysis of languages, forcing digital humanists to justify rather than generate new concepts in working code places. “We are not here concerned with so-called computer 'languages', which resemble human languages (English, Sanskrit, Malayalam, Mandarin Chinese, Twi or Shoshone etc.) in some ways but are forever totally unlike human languages in that they do not grow out of the unconscious but directly out of consciousness. Computer language rules ('grammar') are stated first and thereafter used. The 'rules' of grammar in natural human languages are used first and can be abstracted from usage and stated explicitly in words only with difficulty and never completely” (7). Moreover, referring back to Campbell-Kelly and other software historians, the difficulty of obtaining philosophy written in Africa is like the difficulty of obtaining source code and other documentation of technological undertakings, although the Internet and especially foss practices have reversed this and invites a second look.

We should not be lulled into thinking it is just a matter of retrieving source material for objectivist study or reclaiming a sense of otherness, however, and Janz articulates a similar tension regarding the relationship between written texts and tradition. Instead, he asks “What happens when the reflective scholarly work on tradition is turned back on the culture and becomes part of its life? What happens when philosophy takes seriously the debts and duties it has to the place(s) from which it comes? Under these conditions, we have the potential for new ideas that spring from tradition" (61). Janz's sense of philosophy respecting tradition requiring taking debts and duties seriously is well expressed by the procedural rhetoric of protocol's distributed control operations, and vice versa, working through Galloway, Tanaka-Ishii, and Berry, going beyond emergence from subterranean streams to directedness of technological mastery that is nonetheless peripheral to the philosophical gaze. "Liminality is different in different places, and part of the platial task of philosophy is to identify both the lived meaning of the participants, and also to turn back on itself and recognize its own place in that meaning. This is truly thought thinking itself" (61).

My methodology applies tradition as a mode of thought mediating liminal areas between the rational gaze and its necessary peripheries to philosophical studies of computing and programming, in which embodied thinking necessarily interfaces and potentially programs as it addresses situatedness in places, and tradition includes first and foremost working code encountered in its milieu like `10 PRINT CHR$(205.5+RND(1)); : GOTO 10` running on the Commodore 64 and as found in print magazines from the 1980s.

I have not yet answered the question what are working code places by giving material examples, or discussed how to read and present them. A spatial approach like Berry's survey of types of code provides concise visual examples of poetic, obfuscated, gendered, and beautiful code, for example. A favorite that appears in many texts is Hoare's famous Quicksort algorithm, an exemplar of a brief, recursive procedural function that packs a surprising dynamism that can be discerned by those who learn to read it—if reading is the appropriate term—more like intuit its intended machine operations for various input data sets.[110]

110 I still do not understand how this routine works, although as a youth I managed to copy it from an Apple //

```
void quicksort(int l, int u)
{
      int i, m;
      if(l = u) return;
      swap(l, randint(l,u));
      m=l;
      for(i=l+1; i<=u; i++)
            if(x[i] < x[l])
                  swap(++m, I);
      swap(l,m);
      quicksort(l, m-1);
      quicksort(m+1, u);
}
```

As Jon Bentley describes “The Most Beautiful Code I Never Wrote” in Oram and Wilson's *Beautiful Code* in the context of his thesis on divide-and-conquer algorithms, “I loved the algorithm, but I always tiptoed around its innermost loop. I once spent two days debugging a complex program that was based on that loop, and for years I carefully copied that code whenever I needed to perform a similar task. It solved my problems, but I didn't *really* understand it” (30).

Bentley developed an experimental approach to analyzing the Quicksort algorithm when he noted undergraduates could not get the gist of its mathematical proof, coming around to Knuth's summing factor technique after experimenting with probes in the algorithm code, that is, by making small changes and adding debugging messages to reveal what the machine is doing that is not literally evident by reading the source without sufficient mathematical insight. Thus working code places are where unit operations are encoded, where procedural rhetorics are mounted, and procedural enthymemes string together the stages of software development from conception through implementation and even into deprecation. They are also personal sites of

industry magazine of the sort Montfort et. al. describe, and modify it to work in an Applesoft BASIC program to alphabetize an array of names.

habituation, for both dwelling and sojourning, such as the many versions of GNU/Linux operating systems with which I have done all of my personal and most of my professional computing, excepting mobile devices like my iPhone, which of course invites further self reflection.

Exemplars may be found among the work of philosophical programmers, such as the projects McGann, Bogost, Hayles, Ramsay and others describe, although to present a project without substantial invitation to explore source code situates analysis at the software studies rather than code studies level. It is certainly less meaningful to encounter working code as an observer, fed via textual commentary, than to *dwell* in working code as a practice. Here Janz's notion of platial analysis aligns with O'Gorman's utilization of Lecercle's diachrony within synchrony interpretive style, particularly when deployed via Ulmer's methodological tool *mystory*. The mystory popcycle genre for holistic simultaneous writing potentially combines all of an individual's interpellated discourses. “The core discourses being those of family, entertainment, school (community history), and career (disciplinary field). . . . Mystory assumes that alienation as an experience of dissociation and reification may be overcome in electracy by a practice adopted to the digital capacities of multimedia. This practice is fundamentally aesthetic. Following Lacan's reading of Joyce, mystory brings the discourses into correspondence (uniting them into a popcycle) by means of 'signifiance' (signifierness)—the repetition of a signifier through the details of each discourse, the circulating prop that organizes any well-made narrative, like the letter that passes from character to character in Poe's Purloined Letter” (“Florida Out of Sorts” 30-31). While obviously lending itself to interactive networked multimedia, it also opens the door to core discourses from the scholarly remainder, namely the computer programs written over the course of one's lifetime beginning in childhood. Now punching a black hole in the modern, multiprocessing, high resolution GUI with a Timex-Sinclair 1000 or Apple //e emulator can do more than provide the appreciation of technological heredity and difference that Fuller prescribes. For myself and millions of others who first starting programming as children, the mystory approach to studying working code places is a very tangible and compelling option. The example I have in mind but have not dared peak at yet is my “Student Interest Filing Program” written in Applesoft BASIC for my middle school to automate the quarterly task of signing up hundreds of students for limited seating elective classes based on their first and second preferences.

Working code places are situated, personal, and therefore their presentation in scholarship seldom reaches the sublime plateaus of examples like the Quicksort algorithm, but on the other hand, perform exactly like the original sense of diachrony within synchrony O'Gorman develops. Specifically, were I to look back into the code I wrote as a child for automating my middle school student activity registration process, in which I implemented a quicksort variant for Apple Basic copied from a magazine, clueless as to its actual algorithmic efficacy, and only interested in a faster process than the bubble sort I could derive on my own, those diachronic associations O'Gorman and Ulmer describe may begin to emanate from it. However, a more meaningful use of working code places beyond this gratifying reminiscence ought to be when they are deliberately practiced to form a substantial portion of a scholarly undertaking, such as developing a software project in order to investigate a philosophical question. Practitioners of that method ought to be called programming philosophers.

Programming Philosophers

Manovich's recent ideas regarding modes of cognition and expression, and how they relate to programming practices, feature large in his study of cultural software. With the final publication of *Software Takes Command*, after years lingering as an electronic book, he attempts to reassert the importance of cultural software, applying his historical media studies approach to the specific range of applications like *Photoshop*. “At the end of the twentieth century humans have added a fundamentally new dimension to everything that counts as 'culture'. This dimension is software in general, and application software for creating and accessing content in particular” (*Software Takes Command* 31). Yet he shuns more basic, systems-level applications, which he calls grey software, preferring to focus on mainstream applications used to create and access cultural content. “I am interested in *how software appears to users*—i.e. what *functions* it offers to create, share, reuse, mix, create, manage, share and communicate content, *the interfaces* used to present these functions, and *assumptions and models about a user, her/his needs, and society* encoded in these functions and their interface design” (29). He also makes the deliberate decision not to put his energy into promoting programming, standing by his “commitment to understand the mainstream cultural practices rather than to emphasize (as many cultural critics do) the exceptions, no matter how progressive they may be” (31).

Bogost, on the contrary, recommends philosophers practice a sort of carpentry “constructing artifacts as a philosophical practice” (*Alien Phenomenology* 91), raising hammers instead of fists, akin to O'Gorman's call that they take up soldering irons and build circuits. “While a few exceptions exist (Jacques Derrida's *Glas*, perhaps, or the Nietzschean aphorism, or the propositional structure of Baruch Spinoza's *Ethics* or Ludwig Wittgenstein's *Tractatus*), philosophical works generally do not perpetrate their philosophical positions through their form as books. The carpenter, by contrast, must contend with the material resistance of his or her choses form, making the object itself become the philosophy" (93). These images echo Nietzsche's often misunderstood subtitle to *Twilight of the Idols, or, How to Philosophize with a Hammer*, in which a small tuning hammer is intended to sound out hollow concepts by tapping them, not a sledge hammer to smash them. To be programming philosophers, versus philosophical programmers or traditional philosophers, entails a substantial portion of philosophical thought production to be deliberately done platially with software, not just use that Manovich addresses, but also working code.

An example of philosophical carpentry is Bogost's *Latour Litanizer*, a program that uses the MediWiki API feature for retrieving random news articles "and assembles the results into a list with linked object names, one not dissimilar to the sort found in Latour's writings. . . . Yet the principal virtue of the *Latour Litanizer* is also impossible to reproduce in print: the rapidness and diversity of its results. . . . Not only does the diversity and detachment of being intensify with each fresh litany, but those very qualities also invite further exploration through the link" (95-96). Bogost contends that among the philosophical questions raised by *Latour Litanizer* are choices made in writing code such as a query that filters out "(sexy OR woman OR girl)" in a putatively random search. Taking it a step further, assuming the MediWiki API source code is available, what better place/space to conduct such investigations than in source code comments and differences between revisions?

Hayles is a perennial proponent of programming in the humanities, although not a software developer herself. *Electronic Literature* provides coverage of software projects that enact cyborg sensibilities; *My Mother Was a Computer* stresses the importance of coding technology for humanism; *How We Think* describes a number of software projects that foreground writing programs to conduct scholarly research or answer questions posed by new media. To her, the philosophical questions surrounding computer technology should shift from debates about the status of consciousness compared to machine intelligence to how the human machine cyborg ought to comport itself. “The central question, in other words, is no longer how we as rational creatures should act in full possession of free will and untrammeled agency. Rather, the issue is how consciousness evolves from and interacts with the underlying programs that operate analogously to the operations of code. Whether conceived as literal mechanism or instructive analogy, coding technology thus becomes central to understanding the human condition. . . . In this view, agency long identified with free will and rational mind becomes partial in its efficacy, distributed in its location, mechanistic in its origin, and bound up at least as much with code as with natural language” (*My Mother Was a Computer* 191-192). The activities of humanists and philosophers should likewise shift to include a sizable programming component, which I call working code. “As Manovich says about cultural analytics and Moretti proclaims about distant reading, machine analysis opens the door to new kinds of discoveries that were not possible before and that can surprise and intrigue scholars accustomed to the delights of close reading” (*How We Think* 78).

The Coda to *How We Think* presents in detail how she and Allen Beye Riddell conducted a machine reading of Danielewski's *Only Revolutions*, a text that plays a central role in the concluding chapters as an example of technogenesis redefining the print codex as a digital technology, manifesting aesthetic, neurocognitive and technical implications. By hand coding the entire text into a database “with special categories for cars, plants, animals, minerals, and place-names, and with every word indicated as originating with Sam or Hailey's narratives, respectively. With the place-names identified, we then overlaid them onto a Google map. . . . Further insight is gained by comparing the word frequency of Hailey's narrative with that of Sam's. The extensive parallels between the two narratives are borne out by the correspondence between paired frequency counts” (242). When they discovered that a poster the author provided for his French translator contained a “Nix List” of conceptual clusters and specific words Danielewski did not want to appear in the translation, they sought to discover the words that never appear at all in the work. “Our solution is to compare the word frequencies in *Only Revolutions* with the Brown corpus, a database of one million words carefully selected to be statistically representative of twentieth-century American prose. To assist our comparison, we calculated a chi-square statistic for each word in *Only Revolutions*, which provides a rough measure of how noticeable the difference is between the observed frequencies in *Only Revolutions* and the Brown corpus” (245-245). The result of their analysis was to conclude that the word 'or' itself was excluded from the text. “We conjecture that *or* is forbidden because it is the acronym that Danielewski (and others) typically use for *Only Revolutions*. . . . Thus *self-reflexivity has been banished from the semantic register and displaced onto the topographic*, another indication of how important the spatial aesthetic is to this text” (246).

While this exercise by Hayles and Ridell may seem to be an abuse of university equipment for the sake of solving a manufactured mystery comparable to crowd-sourced spoilers of the TV series *Lost* that Henry Jenkins examines, it bears a striking resemblance to Derrida's study of the Platonic corpus in *Dissemination*. “Curiously, however, there is another of these words that, to our knowledge, is never used by Plato. If we line it up with the series *pharmakeia-pharmakon-pharmakeus,* we will no longer be able to content ourselves with reconstituting a chain that, for all its hiddenness, for all it might escape Plato's notice, is nevertheless something that passes through certain discoverable points of presence that can be seen in the text. . . . The word in question is *pharmakos* (wizard, magician, poisoner), a synonym of *pharmakeus* (which Plato uses), but with the unique feature of having been overdetermined, overlaid by Greek culture with another function" (129-130). We would not want to spoil Derrida's philosophical explanation of this apparent structural gap in the Platonic text, where "some such force, given the system of the language, cannot *not* have acted upon the writing and the reading of this text" (129), by devising software to accomplish similar distant reading techniques as Hayles and Riddell, any more than he questions whether Freud would have thought any differently about psychoanalysis if he had used email—would we?

In the spirit of Janz's platial approach to philosophy, we ought to instead explore the Platonic corpus from the perspective of working it into code. Deleuze and Guattari describe the becoming-other that occurs when visiting such places: "We head for the horizon, on the plane of immanence, and we return with bloodshot eyes, yet they are the eyes of the mind. . . . This is because one does not think without becoming something else, something that does not think an animal, a molecule, a particle and that comes back to thought and revives it" (*What is Philosophy?* 41). What new insights may be discovered can only be documented after the fact, after traces are left that philosophy occurred in these working code places by programming philosophers.

Steven Ramsay provides a hopeful offering with *Reading Machines*, but only gets partial credit for promoting a hacker/scholar as programming philosopher but leaving code work subservient to intuitive humanities thinking ultimately done by humans. As Manovich does with cultural software, he redefines the objective of humanities programming for the service of critical reading strategies, and away from generic control (xi). Algorithmic criticism is already built into reading practices. Echoing the sentiments of Licklider, Kemeny, and Heim, Ramsay views the computer as component of symbiosis to provide computational results for humans to engage in making intuitive inferences. “The understanding promised by the critical act arises not from a presentation of facts, but from the elaboration of a gestalt, and it rightfully includes the vague reference, the conjectured similitude, the ironic twist, and the dramatic turn” (15-16). Yet he does concede that methodological questions of algorithmic textual analysis may be as provocative as hermeneutical ones. He is interested in evaluating robustness of discussion inspired by particular procedures of textual analysis over fitness of the procedures; in Janz's terms, asking *what does it mean to do philosophy in this place?* versus *what are the philosophical conclusions?*

Like McGann's deformation experiments with scanning Rossetti's images, Ramsay's method hinges on developing software that reads texts in new and potentially unexpected ways. “Algorithmic criticism therefore begins with the machinic inflection of programming—a form of textual creation that, despite the apparent determinism of the underlying machine, proceeds always in organic and unexpected ways. . . . It is by nature a meticulous process, since to program is to move within a highly constrained language that is wholly intolerant toward deviation from its own internal rules. But the goal of such constraint is always unexpected forms of knowing within the larger framework of more collective understandings. . . . The hermeneutics of what is becomes mingled with the hermeneutics of how to" (63). While prior hermeneutic understanding is required to develop programs for textual deformation in the first place, he toys with differences between a run once arrival at a deformation to interpret versus constantly operating under the condition of reciprocal transformation between programs and texts. "But in another sense it is a recursive process. In order to understand the text we must create another text that requires an understanding of the first text" (66).

Ramsay's short book rapidly concludes with a chapter called *'Patacomputing*, referencing Alfred Jarry's pataphysics, the humorous science of imaginary solutions and apotheosis of perspectivalism often proposed as a model for the inverted science of humanities computing, whose object is to create questions more so than answer them. There he attacks a number of popular textual analysis software applications that advertise themselves as providing methods and procedures for algorithmic criticism, but ultimately just provide yet another set of tools for searching. "Again and again, the language of *TAPoR* [Text Analysis Portal for Research] points not to methods or procedures, but to tools—things to be wielded against any text on the Web (the default examples optimistically include both a corpus of French medieval poetry and the *Universal Declaration of Human Rights*). . . . Text analysis of the sort put forth by *WordHoard*, *TAPoR*, and *HyperPo* suggests other antonyms to close reading, including what Franco Moretti has called distant reading" (74-77). While Hayles is more measured in her critique of algorithmic deformation, Ramsay argues this breed of tools produces what Derrida called overpotentialized text. "What is different about digital archives is the way in which text analysis procedures (including that most primitive of procedures: the keyword search) has the potential to draw unexpected paths through a documentary space that is distinguished by its overall incomprehensibility. . . . The result of a system like *MONK* [Metadata Offer New Knowledge] is the same as that for virtually any text-analytical procedure: a textual artifact that, even if recapitulated in the form of an elaborate interactive visualization, remains essentially a list" (78-80).

Ramsay concludes that computational practices may not become critically tractable until they are also commonplace, when the hacker/scholar are not mutually exclusive (80).[111] "Those activities that are usually seen as anathema to the essential goal of literary criticism—quantitative analysis chief among them—will need to be reconsidered if it turns out that backward poems lie at the root of our forward endeavors. Our fear of breaking faith with the text may also need to give way to a renewed faith in the capacity of subjective engagement for liberating the potentialities of meaning" (57). However, he also suggests that "it may be that the tools of algorithmic criticism are like Wittgenstein's ladder. When we have used them to climb up beyond, we recognize them as nonsensical and cast the ladder aside" (80). After critical use of computational practices have become ingrained in humanities scholars, "the bare facts of the tools themselves will seem, like the technical details of automobiles or telephones, not to be the main thing at all. . . . For by then we will have understood computer-based criticism to be what it has always been: human-based criticism with computers" (81). Curiously, criticism like McGann's evolving from reflecting about evolution of an XML schema for creating an electronic archive or electronic scholarly edition does not seem within the scope of Ramsay's algorithmic criticism, although estrangement, defamiliarization, and deformations produced by software are.

111 Algorithmic criticism provides activities for hacker scholar besides toiling with TEI and XML, for instance.

Like Hayles, McGann opens the scope of theory to potential new concerns and techniques to investigate them. “How to incorporate digitized images into the computational field is not simply a problem that hyperediting must solve; it is a problem created by the very arrival of the possibilities of hyperediting. . . . Those of us who were involved with The Rossetti Archive from the beginning spent virtually the entire first year working at this problem” (69). We can appreciate his detailed account of theory-informed and exploratory technological development as a prototype for critical programming, for he provides the details of his software project at the level of designing “a structure of SGML markup tags for the physical features of all the types of documents contained in The Rossetti Archive (textual as well as pictorial); and second, to develop an image tool that permits one to attach anchors to specific features of digitized images" (69). He recognizes instantiated arguments are like creating software projects to do humanities research, what William Carlos Williams called the embodiment of knowledge: “they call attention to the theoretical opportunities involved in making an edition. The totalized factive commitments and obligations of an editorial project open into a theoretical privilege unavailable to the speculative or interpretive essay or monograph” (80).[112]

In this sketch of philosophical programmers I have reserved the top position for McGann because he not only deliberately creates programs to conduct humanities work, but he also foregrounds awareness of the iterative process of that working code impacting the guiding thoughts of his methodology, embodying what he calls poiesis-as-theory. Like bricoleur programming versus hard mastery in Turkle, different from unknown knows of psychoanalysis

112 He seems to be taking the designer, system-centric perspective: could the flip side of the quest for designing ever more decentered tools be to foster user involvement in creating the interfaces?

as Hayles points out, he differentiates poiesis-as-theory as instrument or machine making engineering projects from traditional theory, comparing it to the innovative demands of artistic work. “The close relation it bears to artistic work is important because poiesis-as-theory makes possible the imagination of what you don't know. Theory in the other sense for instance, Heideggerian dialectic is a procedure for revealing what you *do* know but are unaware of” (83). Noticing his own habit of taking undisciplined guesses, hacking, to solve the problem of tagging digital images causes a particular philosophical problem to become apparent. “The commitment of *The Rossetti Archive* to elucidating digital images kept generating one of its most important series of logical impasses and failures (for more on this, see the appendix). The problem—images recalcitrance to analytic treatment—meant that we were continually kvetching over the matter, which in turn meant that I would often find myself engaging the problem in undisciplined ways and circumstances” (83-84).

It is not difficult to hear echoes of Heidegger's famous study of *noein* and *legein* as modes of knowledge as McGann ponders the needs of images versus sequential character texts. “One kind of project is presentational, designed for the mind's eye (or the eyes' mind); the other is analytic, a logical structure that can free the conceptual imagination of its inevitable codex-based limits. The former tend to be image-oriented, the latter to be text-based” (88). Thus the appendix in the chapter he claims is the center of the book in terms of the chronological development of his thoughts about radiant textuality is also where I make the most connections to my own research, not only the image/text and surface/depth distinction, but also the use of revision history and source code comments to trace the evolution of theoretical thought embodied in poiesis, places where philosophy occurred.

McGann's account of interactive, iterative bootstrapping development is familiar to any bricoleur software developer. “Our plan was to use the construction process as a mechanism for imagining what we didn't know about the project. In one respect we were engaged in a classic form of model-building, whereby a theoretical structure is designed, built, and tested, then scaled up in size and tested at each succeeding junction. The testing exposes the design flaws that lead to modifications of the original design” (91). While he initially did not pay attention to this change history, he later realized he could review the consecutive code submissions whose 'delta' of changes revealed what he later looks back on as his evolving, unstructured meditation on the recalcitrance of images to by read like searchable texts. “That process of development can be illustrated by looking at one of our SGML markup protocols—the DTD for marking up every Rossetti archive document (or RAD). . . . My interest here is not in the SGML design as such but in the record of modifications to the design. That record appears as the list of dated entries at the top of the document. . . . A great many modifications to the initial design were made during that year, but we did not at first think to keep a systematic record of the changes” (91).

Theorizing via revision history and comments, as I am doing in software source code, is the big insight he and I both want to leverage as an advance in humanities scholarship emerging from engaging in computer technologies as producers. He admits that not enough comments were made in those early iterations; “the record does not indicate certain decisive moments when the archive was discovering features of itself it was unaware of. In these cases no actual changes were made to the DTDs” (91). He seems to be attributing a cognitive role to the evolving archive as an information technology integration exercise, following Hayles' human-computer cyborg articulated in *Electronic Literature*.[113] What came to make sense as iterative changes to the protocols that made the system work better in retrospect reflect the discovery of unknowns as if the result of Socratic self-questioning.

Where McGann's working code matters the most to me is how he recognizes future work and intellectual problems revealed by comments made of their iterative development of DTD tags constituting the history of RAD revisions. He writes, for example, “look at the notation for 14 June 1995: `<!-- revised: 14 Jun 95 to add group to rad for serials -->` A large-scale change in our conception of the archive's documentary structure is concealed in this small entry. That practical insight, however, was not nearly so interesting as the insights we gained into general problems of concurrency and into the limitations of SGML software” (93). Collating units of prose texts, general problems of concurrency, limitations of SGML software, and text itself are exposed by working code. What is left is to make this accidental means of discovery into a method.

Critical Programming

In the introductory chapter I proposed the collective intelligence problem of our era sees humans settling into a suboptimal intellectual comportment with respect to machines that continue to improve in comparison. A hidden reason may be that philosophers of computing

113 Another example McGann gives hinges on the fact that Rossetti himself paid great interest to taking photographs of his own work, which opened substantial questions regarding intellectual property that anyone making a web site understands today. Leveraging out of copyright photographs Rossetti took of his own paintings solved an economic problem and invited new theoretical speculation. The digital media corollary has been interest in free, open source software and creative commons licenses.

from Manovich to Bogost permit playing games and manipulating interfaces to serve as substitutes for the quintessential representative of procedural rhetoric that is programming. It is crucial to note that Bogost, at least in his earlier work, focuses on human oriented rhetorical domains, whereas the path I steer is right into the inner workings of machines with the rhetorical outcome of instilling programming as problem solving and even a way of conducting humanities research (*Persuasive Games* ix). Why is it worth revisiting studies on learning programming with wider scope in which play itself, and by extension ancillary behaviors to programming, have procedural learning functions? Because the subsumption of explanatory levels implied by the diachrony in synchrony model harbors the implicit danger of obfuscation when the lossy dip into lower levels is reversed and comes back to the upper levels. *The quintessential programmed vision is that double mediation is faithfully represented by direct manipulation*.

Turkle refers to it as a tale of two aesthetics but they are basic and significantly different epistemological positions. “If my transparent Apple II modeled a modernist technological aesthetic, the Macintosh was consistent with a postmodern one. . . . With the introduction of Microsoft Windows in 1985, the modern and postmodern aesthetics of computing became curiously entwined. . . . There is still a tendency to assume that the choice of operating systems is a purely technical decision” (36; 41). In the terms of underground streams that Berry uses, what subsumes beneath the interface does not necessarily bubble back to the surface without transformation. This idea is illustrated by the introduction of noise into the Barszcz C recipe in the Software Studies lexicon by using a preprocessor directive to define `ingredient` as the `char` data type, for the want-to-be template function `ingredient **take()` returns a native unit data type `char` (character), which becomes a decontextualized copy of a character, not a pointer into the contextual situation in which it momentarily shimmers. The mistake is to take human machine borders as atomizing, decontextualizing boundaries rather than as close fitting forms such that the human and machine respect each other's idiosyncratic cultural habits, comparable to Hayles use of Bourdieu's nonsymbolic knowledge repositories, and a second self to Turkle.

Haraway's suggestion that our technological milieu is a coding trickster rather than a neutral database invites studying simulacra technosciences like electronics, computer programming, and digital communications as good introductory exercises. “Perhaps our hopes for accountability in the techno-biopolitics in postmodern frames turn on revisioning the world as coding trickster with whom we must learn to converse. . . . Coyote is not a ghost, merely a protean trickster” (209). Bogost supports a Kittler and Postman inspired entry to understanding machine embodiment, calling for procedural literacy training, hearkening back to Turing's “specter of a totally programmable world. . . . No matter the talent or manipulation of the human programmer, human experience of the machine (which must always be mediated by software) is limited by the architecture of the hardware. . . . Because human relations with the computer is software-based, we need to understand technology's own agency, and how hardware relates to particular software packages” (*Unit Operations* 36-37). Imagine doing unit analysis on early 8 bit, non internetworked computer games as Bogost does so masterfully with traditional literary studies, interpreting literary texts metaphorically as programming. The degree of depth in critical functions of modern games that is practiced in philosophical study of past generations of computing machinery (platform studies) that Bogost performs with modern, web based games, can then be carried over into personal software projects.

In *What is Philosophy,* Deleuze and Guattari explain why philosophy needs science—science is coextensive with contemporary language. “If philosophy has a fundamental need for the science that is contemporary with it, this is because science constantly intersects with the possibility of concepts and because concepts necessarily involve allusions to science that are neither examples nor applications, nor even reflections” (162). Bogost has demonstrated how videogame criticism provides a place for doing philosophy by extending the natural scope of procedural literacy from programming to game play and interface usage in general. His optimistic conclusion in *Unit Operations* is that videogame criticism instantiates Badiou's form of thinking not just in texts but in the world, and can rejuvenate the university, inspiring dreams of new types of videogames produced by dissolving organizational divisions, especially between for profit and not for profit organizations, and different educational institutions.

Critical programming studies are enacted because philosophy likewise needs technology as it once always melded with neighboring science and originally rhetoric, and for an additional reasons beyond the responses that have been given by the theorists on whose shoulders I stand. It is specifically the demise of learning programming as an everyday component of socialization in technological societies, and the substitution of programming skill with competency using the interfaces of cultural software, so that even playing games has been judged an appropriate transference of the psychic energy once destined for working code. Recall Kemeny's rationale for inserting BASIC programming in the general Dartmouth curriculum in the late 1960s: “the students learn an enormous amount by being forced to teach the computer how to solve a given problem. . . . The student must concentrate on the basic principles; he must understand the algorithm thoroughly in order to be able to explain it to a computer” (79).

Kemeny was overly optimistic that programming would remain sufficiently valorized once sound, intuitive, user friendly interfaces arose in massive software projects like Microsoft Windows, Apple OS, and even GNU/Linux foss. Instead, as Manovich argues via his resurrection of Bruner's multiple learning dimensions in Kay's early Dynabook experiments, teaching the computer how to solve a given problem occurs in iconic and enactive dimensions as well as at the symbolic level. “According to Kay, the key step for him and his group was to starting thinking about computers as a medium for learning, experimentation, and artistic expression which can be used not just by adults but also by children of all ages” (97). To appeal to children of all ages meant to simplify the interfaces by making them user friendly, and replace a number of operations that were routinized by obscure symbolic instructions with direct manipulation. “Kay's interpretation of this theory was that a user interface should appeal to all these three mentalities. In contrast to a command-line interface, which is not accessible for children and forces the adult to use only symbolic mentality, the new interface should also make use of emotive and iconic mentalities” (98). Media must be thought beyond symbols; we are on the right track correcting the ideology of direct manipulation, in which the medium disappears, with positions leveraging material specific affordances of media.

Thus there remains a hidden reason why removing the need to program from the interface by altering it from a command and obey dialogue between human and machine to a navigable space unintentionally weakened human intelligence. Double mediation implies the desktop icons are not so much linguistic levers as Rube Goldberg contraptions. Bogost aptly describes them in *Alien Phenomenology.* “Unlike redwoods and lichen and salamanders, computers don't carry the baggage of vivacity. They are plastic and metal corpses with voodoo powers” (9). The problem is that intentionality embodied in deliberately arranged programming instructions can descend into the machinic and resurface largely intact, whereas interactions that transpire as unit operations governed by the interface level, so-called direct manipulation—a mouse click, menu selection, or other action triggering built in code—always return as programmed visions.

Rhetoric professor Robert Cummings realized that teaching programming aids general writing skills because “at its heart, the comparison between writing and coding is instructive to both pursuits because the work product of both groups holds meta-cognitive sway over the thinking processes that create text. That is to say that both writer and coder are moved by the manner in which their text, once written, impacts the thinking process that composed it” (432). Applen and McDaniel propose a theorist-practitioner role that combines technical and humanities competencies, with emphasis on leveraging custom code to explore and meet overall requirements derived from rhetorical analysis, concluding that “we need to recognize that by relying on pre-existing parsers, our creative potential and expressive capacities are limited by the designs of other companies or other individuals. Only by immersing ourselves in the low-level programming of XML parsers can we truly design an interactive system for dealing with XML code in exactly the way we want” (294). Montfort et. al. declare one key software studies method is writing programs to interpret other programs. “It takes this approach to the extreme and builds a large program, using 10 PRINT as the starting point. Just as literary scholars study a text by generating more texts, it is productive to study software by coding new software. In this particular case, it's possible to develop a series of hermeneutic probes in the Commodore BASIC—probes of increasing complexity, programs that transform 10 PRINT's output into a stable, navigable, and testable maze” (244).

The generate and test method from computer science can be deployed by critical programming for studying humanities problems; it leverages the ability of simulations to be generated and submitted to testing in ways impossible, unethical, or cost prohibitive in physical correlates. Consider reasons for distinguishing writing programs as hermeneutic probes with merely using other software to study software, continuing the argument stemming from Kemeny on the value of learning by being forced to teach the machine how to solve a problem: this is the purview of critical programming. Make programming a key competency, not merely an aid like learning a foreign language to better understand the grammar of one's mother tongue, and then see what may be found in these places to motivate inquiry.

CHAPTER 4: CRITICAL PROGRAMMING STUDIES

This chapter explores how philosophy happens in the working code places articulated in the methodology proposed in the previous chapter, delving into three software projects that I have been developing for the past two decades as sites for expanding my philosophical horizons in the context of my professional career as a software engineer and graduate student in the Texts and Technology program at the University of Central Florida. Forming the synthesis portion of this dissertation, I will offer my own foss triad of projects `symposia`, `tapoc`, and `pmrek` as examples of critical programming studies that intentionally parallel and remediate orality, literacy, and electracy. They combine deep dives into program source code with meditations on how these efforts further humanities research by presenting new places for thought: a virtual reality, aural-centric version of Plato's *Symposium*, a self-compiling dissertation writing machine, and a no-holds-barred exercise in electronics, low-level hardware interfaces, operating system kernel module design, and process control system programming to help intuit what machine cognition may be like.

Symposia Remediates Orality

The three foss projects I am reviewing to demonstrate critical programming in practice intentionally play on a number of themes introduced throughout the dissertation, showing how philosophical questions can be cast, queried, and responses discerned through ongoing, iterative working code revisions as well as being enacted by the running of that very code. Each project parallels one of the periods of human intellectual history from the threesome orality, literacy, and electracy by exercising the relevant phenomenal registers.[114] Thus the first project, `symposia`, has a substantial and essential aural component—it is listened to as much as it is read. Symposia are the plural of symposium, which today would be thought as collection of academic events, perhaps, but literally a number of Greek drinking parties, leading back to the most famous drinking party in all of Western philosophy, the one recorded by Plato, if we are to take its self-reported lineage as factual. While philosophers of computing and textuality scholars often cite Plato's *Phaedrus* as a philosophical reflection on the invention of writing as the new medium of its time, I have always considered *Symposium* as a forerunner of meditations on virtual reality—the basic sense of immersive multimedia. This is because the work purports to be a transcription of a conversation between two individuals taking the long walk between Athens and Piraeus, during the course of which a second account is given of a drinking party held years prior featuring Socrates leading a series of speeches about love.

When we silently read the text, we may take some liberties to account for the different personalities as we subvocalize their speeches, but do we give any consideration of their physical arrangement at the banquet? If the premise of the story holds that Aristodemus witnessed the whole thing and has been repeating it from memory—what other tools does he have?–does it mean anything that he may have had trouble hearing a speaker farther away, and thus repeats that speech with less detail and vivacity than another who uttered it closer to him? There are certainly a number of possible physical arrangements of the actual event the text presumes to record, even if we do not believe it is really honest, but instead a literary simulation. There can be at least two

114 The assignment of electracy as the name for the present period is provisional, accepting Ulmer's term without all of his accompanying theory.

basic layouts of the *Symposium* speakers, straight line or angled seating (in cases in which Agathon has a large table; there could have been multiple small groups, although the layout does not change much besides allow Aristodemus to be situated equidistant from all the speakers for perfect fidelity and accurate speech to text conversion). The original text suggests a likely layout of the position of the speakers relative to where the primary listener and recorder of the event, Aristodemus, was located. The speech of Phaedrus, the furthest from his ears, is the least distinct, and those of Agathon, Socrates, and Alcibiades are the most.

Figure 2 presents a number of hypothetical layouts of the speakers in the *Symposium*, both the external frame of the imaginary walk during which the story is told, and the arrangement of participants in the drinking party itself. In the legend of speaker numbers, three (3) is the reader, the companion who has a voice in the text to form the outer frame.

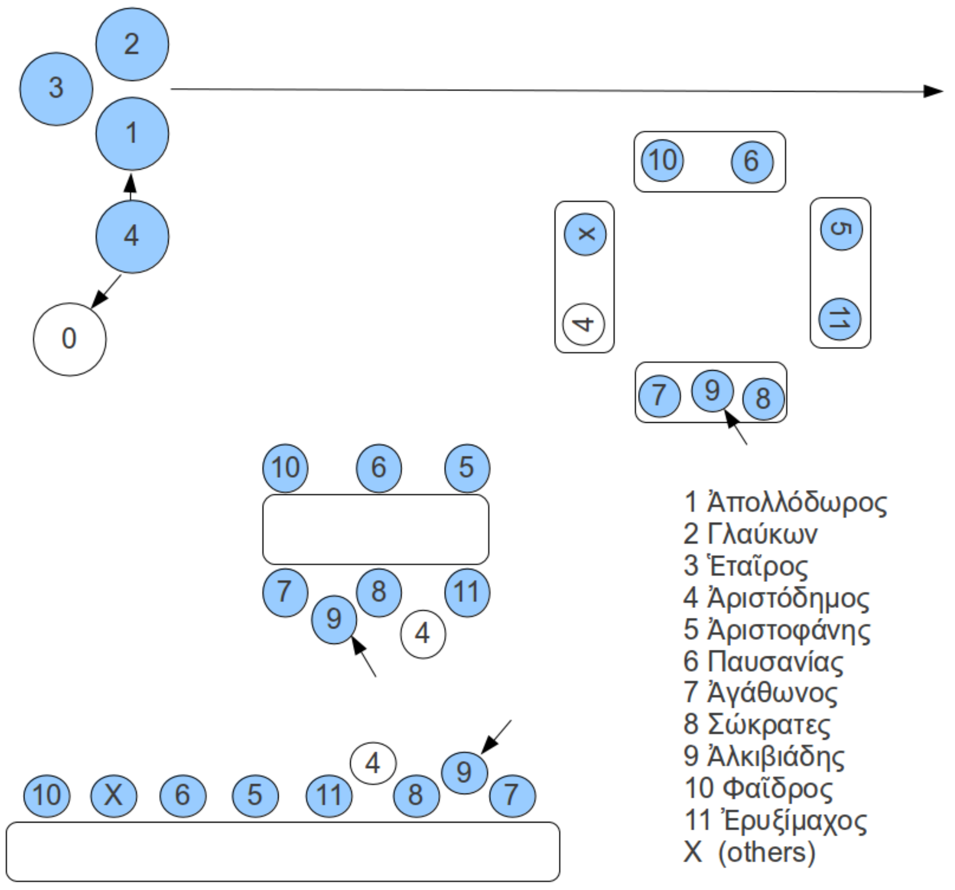


Figure 2: Hypothetical Physical Layouts of Speakers

It was a very simple thought I had back in the early 1990s that launched a second bachelors degree in electronics and computer science, and my subsequent career as an electronics technologist and eventually software architect of process control solutions (I work in Manovich's grey software). As a philosophy graduate student, when I first read *Symposium*, and was spending too much time playing the first person shooter video game *Wolfenstein 3-D*, dabbling with analog video recording equipment before personal computers made digital video editing an everyday consumer practice, I had the thought: what would it be like to experience this text from the perspective of a first-person video game, huffing and puffing walking beside Glaucon as the unnamed companion, then sitting next to Aristodemus during the speeches? How does reading while walking alter subjectivity? The active listener in the symposium proper is Aristodemus: what did he actually hear that night? First and foremost, the experience would be audible. Would it transpire in Greek? How would the voices of the different characters sound? What missing dialogue may be heard in the gaps of versions handed down from antiquity? What other clues embedded in the written text might influence a real virtuality production?[115]

Sadly, the technology was not available at the time to do anything more than imagine it; I lacked the programming skill and the components did not even exist to hack together via the extant cultural software. This vision remained with me for nearly twenty years until the opportunity arose in the context of graduate coursework in sound studies—Tony Grajeda's fall 2011 audio texts and technology seminar. Furthermore, an adequate set of supporting applications and an operating system to enact the speech synthesis of ancient Greek could now be readily assembled using free, open source software available on the Internet. The Greek text itself was freely available from a number of sources, notably the Perseus Project, which had been in its infancy when I first pondered a software version of the *Symposium*. However, this project would not have been feasible without the `espeak` format synthesis foss project that offered a C library API that could produce ancient Greek audio output from a text string input.

115 I did not have Castells' term in my vocabulary at the time, but I likewise understood what I was imagining as a means of describing the programmed synthesis of simulacra for real time perceptual experience by human beings.

The initial specification of the software project before any code was written was overly optimistic, as are most initial specifications.[116] This proposal outlined a software project that integrates text encoding, speech synthesis, and the creation of multiple, concurrent auditory fields within the virtual space of an artificial world. Through it, Plato's *Symposium* can be recast as a boisterous, computer-generated cacophony that humans experience as an audible phenomenon, rather than a passive linear sequence of written characters whose audible characteristics must be imagined or vocalized while being visually read. Made as a free, open source software project entirely of components that are also foss, it can be released to the global community via Sourceforge so as not to depend on my future tending to keep it alive.

I gave the project the provisional name `symposia` to emphasize the multiplicity of media, speakers, and listening positions that will be actualized in the virtual reality environment, in addition to paying homage to my abortive attempts in the early 1990s to meld philosophy and programming before its time. The technical requirements (and wish-list) included: text to speech in various languages, encoding of pacing and other audible features that are implicit in sound reproduction in source text, spatialized generation of speech and other sounds with respect to the current position of the listener within the auditory field, traversal of the listener in the auditory field, 'filtering' of 'unmediated' source text based on implied characteristics of the virtual listener (age, social status, degree of intoxication, locus of desires, and so on), and delivery of final audible production in an immersive physical or virtual reality environment.

116 The history of computing and software is full of overly optimistic initial project proposals that wind up taking much longer to produce much less.

Table 2 presents the primary components of `symposia` in terms of the custom code I wrote that eventually made it into the Subversion repository described below.

Table 2: Source Code Files for Symposia

Source File	**Language**	**Functional Description**
helper_scripts.sh	BASH	Scripts called by Makefile
index.hsml	HTML	Sourceforge.net project home page
Lexia.sql	MySQL SQL	Table dump of Symposium Greek text
Makefile_generic	GNU make	Makefile to build project
README.txt	plain text (English)	Installation and configuration instructions
Speaker.sql	MySQL SQL	Table dump of Symposium speakers
start.console.sh	BASH	Script to start symposia-console
start.symposia.sh	BASH	Script to start a symposia speaker instance
symposia-console.cpp	C/C++	Control console
symposia.cpp	C/C++	Program for symposia speaker instance
symposia.h	C/C++	Header file of default values
symposia.php	PHP	Web admin utility

Because it links in other libraries like `espeak` for speech synthesis and `mysql` for the relational database functions, its true footprint extends into the surrounding operating system. This point is important, for even the one-liner `10 PRINT` extends far into its platform to give it meaning. Furthermore, it foregrounds what may be called the *vicissitudes of workspace preparation*, the often extensive set up before the first lines of code can even compile.

A program like this was unimaginable with 1990s consumer programming capabilities, especially a speech synthesis utility that handles ancient Greek, and the seamless integration of Greek text into the MySQL relational database's storage engine after being copied from a web browser at the Perseus website. Thus an initial task was selecting and installing a speech synthesis development library. Searching Sourceforge.net, I found `espeak` fit the criteria, but a number of other operating system packages were required just to use its compile-in library functions gotten with the `#include “speak_lib.h”` preprocessor directive, requiring the aforementioned workspace preparation. Other programmers have their “Hello World” programs; `symposia.cpp` begins by pronouncing the opening line of Plato's text.

```
#include "speak_lib.h"
espeak_ERROR espeak_error;
espeak_POSITION_TYPE espeak_position_type = POS_CHARACTER;
string initial_utterance = "nothing";
unsigned int synth_flags = espeakCHARS_AUTO | espeakPHONEMES |
espeakENDPAUSE;
initial_utterance = "δοκῶ μοι περὶ ὧν πυνθάνεσθε οὐκ ἀμελέτητος εἶναι.";
. . .
espeak_error = espeak_Synth(initial_utterance.c_str(),
                             (int)initial_utterance.length()+1,
                             0,
                             espeak_position_type,
                             0,
                             synth_flags,
                             NULL,
                             NULL);
```

The vicissitudes of workspace preparation no doubt also have an impact on the shape of later research thinking. On the positive side, once a program begins working, after the seemingly interminable initial preparatory work, it reads itself—pun intended—and takes on a life of its own, encouraging further development.

At the heart of the *Symposium* is the sequential sequence of utterances by eleven discrete speakers, each with their (they were all men) combination of characteristics that make what Barthes calls the grain of the voice. Very crude (and male) by default, the initial utterance nonetheless begins something that has never existed in philosophy for thousands of years, the ability to make a virtual reality presentation of Plato's *Symposium* that is first heard in its native

Greek via formant synthesis, real simulacra.[117] The how to do it was complicated and daunting but nonetheless of manageable complexity, and a solution soon came together using the MySQL database to store individual utterances by speaker, sentence by sentence. The table name `Lexia` was deliberately chosen to designate these unit operations as small textual units, honoring Talan Memmott's work of electronic literature *Lexia to Perplexia*, a favorite of Hayles and other digital media scholars.

```
CREATE TABLE `Lexia` (
      `id` int(11) NOT NULL AUTO_INCREMENT,
      `PHI` float NOT NULL,
      `Position` double NOT NULL DEFAULT '0',
      `Label` varchar(32) NOT NULL,
      `Writing` text NOT NULL,
      PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1000021 DEFAULT CHARSET=utf8;
```

Even this table schema bears the marks of its revision history. The first version used the automatically-incrementing integer `id` as the key. Later I added the two identifiers `PHI` to designate the speaker (based on the list in Figure 2), and `Position` to situate the utterance in the overall span of the performance. The lexia itself is stored in the `text`-typed field `Writing`, which upon setting the `DEFAULT CHARSET` to `utf8`, was able to store Greek character strings copy and pasted from the Perseus website. The use of floating point identifiers rather than integers came about as a solution to implementing an idea inspired by our readings in the seminar about polyglot reading practices performed in 1916, in which multiple speakers recited the same poem in different languages simultaneously (Kahn 48-49). I wanted the software to be able to pronounce an English translation alongside the less familiar Greek as an experiment that

117 In the classification of speech synthesis methods, formant synthesis generates sounds from native machine code versus concatenating recorded vocal samples.

symposia could underwrite. By altering the volume and time delay between the primary Greek and secondary English voices of the same lexia, an entirely new form of polyglot experiments could commence. The crude solution typical of my bricolage programming style was to use the tenths decimal place to signify the language of the speaker (PHI) and utterance (Writing), as shown in Table 3. Zero-tenths designate the default language, ancient Greek, and five-tenths designates English.

Table 3: Encoding Polyglot Utterances in the Lexia Table

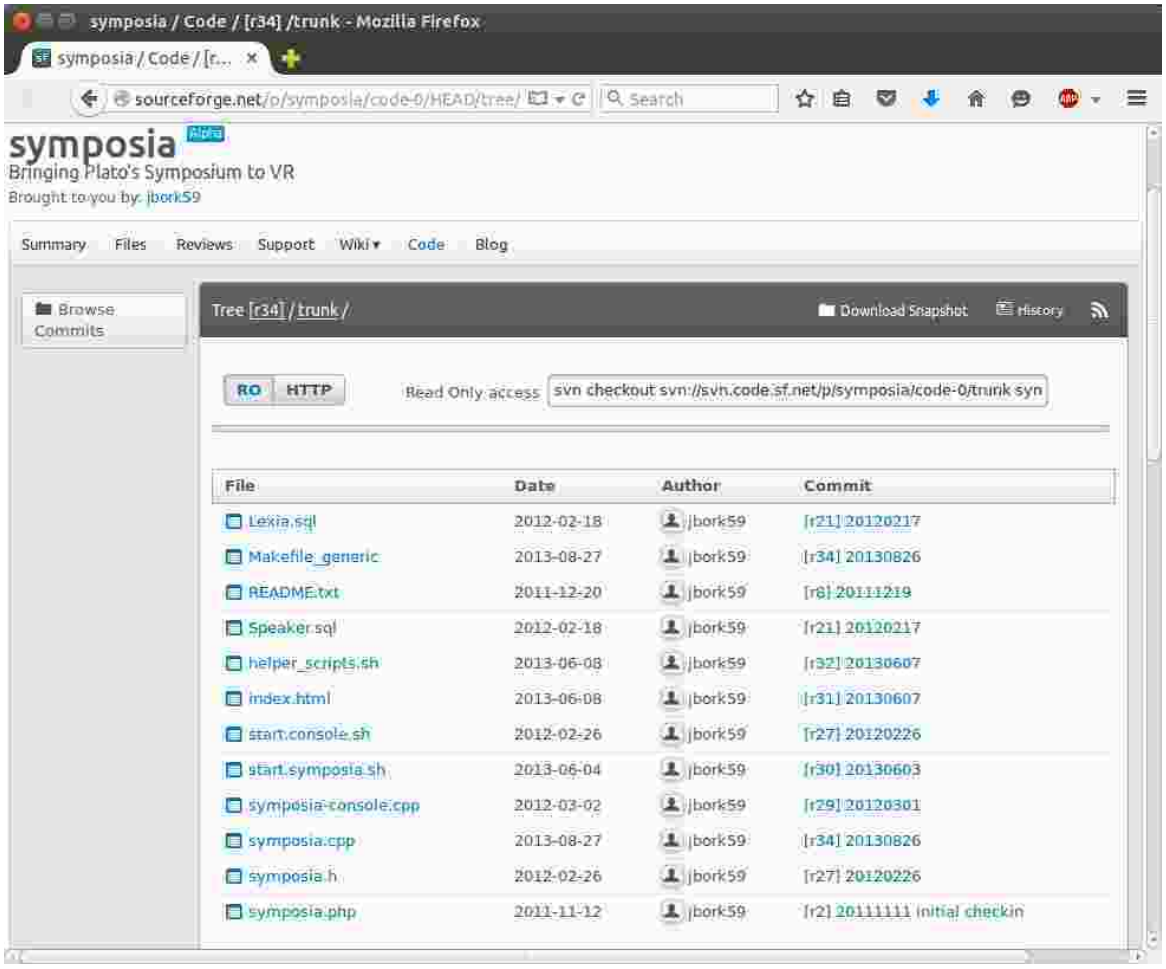
PHI	Position	Writing
1	1	δοκῶ μοι περὶ ὧν πυνθάνεσθε οὐκ ἀμελέτητος εἶναι.
1.5	1.5	Concerning the things about which you ask to be informed I believe that I am not ill-prepared with an answer.
1	2	καὶ γὰρ ἐτύγχανον πρῴην εἰς ἄστυ οἴκοθεν ἀνιὼν Φαληρόθεν:
1.5	2.5	For the day before yesterday I was coming from my own home at Phalerum to the city,
1	3	τῶν οὖν γνωρίμων τις ὄπισθεν κατιδών με πόρρωθεν ἐκάλεσε, καὶ παίζων ἅμα τῇ κλήσει,
1.5	3.5	and one of my acquaintance, who had caught a sight of me from behind, calling out playfully in the distance, said:
2	4	‘ὦ Φαληρεύς,
2.5	4.5	O thou Phalerian
1	5	ἔφη,
1.5	5.5	he said,
2	6	‘οὗτος Ἀπολλόδωρος, οὐ περιμένεις;
2.5	6.5	Apollodorus, halt!
1	7	κἀγὼ ἐπιστὰς περιέμεινα.

PHI	Position	Writing
1.5	7.5	So I did as I was bid;
1	8	καὶ ὅς,
1.5	8.5	and then he said,
2	9	Ἀπολλόδωρε,
2.5	9.5	Apollodorus,
1	10	ἔφη,
1.5	10.5	he said,
2	11	καὶ μὴν καὶ ἔναγχός σε ἐζήτουν βουλόμενος διαπυθέσθαι τὴν Ἀγάθωνος συνουσίαν καὶ Σωκράτους καὶ Ἀλκιβιάδου καὶ τῶν ἄλλων τῶν τότε ἐν τῷ συνδείπνῳ παραγενομένων, περὶ τῶν ἐρωτικῶν λόγων τίνες ἦσαν:
2.5	11.5	'I was looking for you only just now, that I might ask you about the speeches in praise of love, which were delivered by Socrates, Alcibiades, and others, at Agathon''s supper.'

As this project evolved, so did my practices for source code maintenance. Like the beginnings of most personal coding projects, it began by making a subdirectory on a particular computer and creating files, with no thought of source control. They were saved as they were edited, but the progression of changes was not concretized in formal revisions. Only after some progress had been made did I create a Subversion repository for it and do an initial check-in. Therefore, the early revision history is not readily reconstructed without searching for comments in the various code modules where I had the presence of mind to add a 'timestamp' that includes the year, month, and day of the change. The modification to `symposia.cpp` shown below allows the language of the speaker to be changed from the default ancient Greek, as well as some aspects of the 'grain of the voice', based on tunable parameters offered by the `espeak` application program interface (API) beyond the rate, volume, and pitch that I had initially coded.[118]

```
/* this is the one being used by the process() loop */
int ensound(std::string lexia, int rate, int volume, int pitch, int range,
char * voice, char * language, int variant, int age, bool block)
{
      . . .
      espeak_ERROR espeak_error;
      espeak_VOICE voice_select;
      . . .
      if(just_initialized)
      {
            /* 20120227 add variant to language specification (always male
voice) */
            if(language || variant || age)
            {
                  if(language)
                        voice_select.languages = language;
                  if(variant)
                        voice_select.variant = variant;
                  if(age)
                        voice_select.age = age;
                  fprintf(output, "ensound: language [%s] gender [1] variant
[%d] age [%d]\n", language, variant, age);
            }
            else
                  voice_select.languages = DEFAULT_LANGUAGE;
            /* this is the next important thing to do */
            espeak_error = espeak_SetVoiceByProperties(&voice_select);
            close(2);
      }
```

118 Barthes distinguishes between pheno-song and geno-song qualities of vocal speech to propose the latter characterizing the 'grain of the voice', namely the volume, spacing, diction of the language, “the space where signfications germinate 'from within language and in its very materiality',” in contrast to the “features which belong to the structure of the language being sung, the rules of the genre, the coded from of the melisma, the composer's idiolect” (*Image-Music-Text* 182).



symposia / Code / [r34] /trunk - Mozilla Firefox

sourceforge.net/p/symposia/code-0/HEAD/tree/

symposia Alpha
Bringing Plato's Symposium to VR
Brought to you by: jbork59

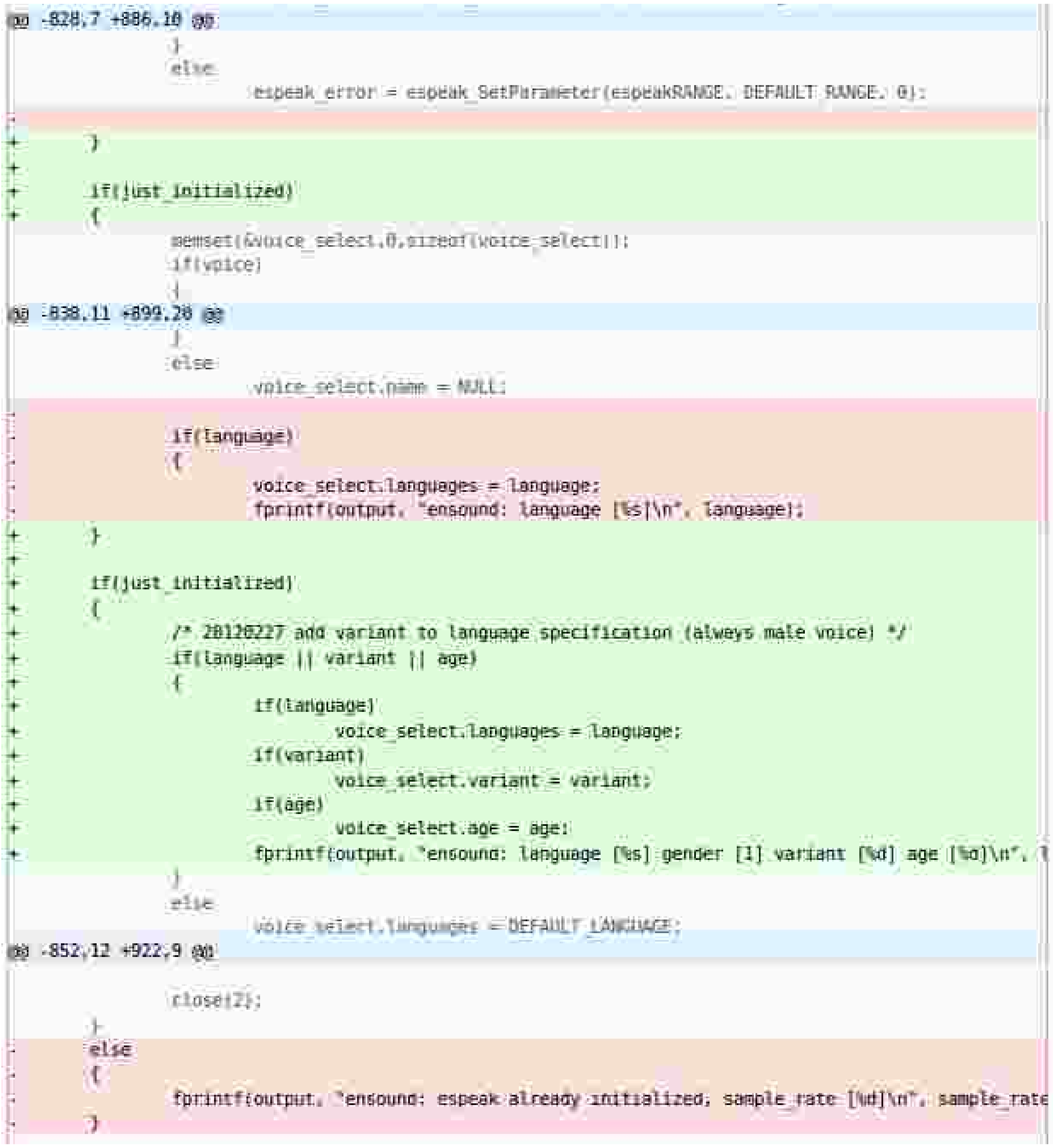
Tree [r34] / trunk /

Read Only access: svn checkout svn://svn.code.sf.net/p/symposia/code-0/trunk syn

File	Date	Author	Commit
Lexia.sql	2012-02-18	jbork59	[r21] 20120217
Makefile_generic	2013-08-27	jbork59	[r34] 20130826
README.txt	2011-12-20	jbork59	[r6] 20111219
Speaker.sql	2012-02-18	jbork59	[r21] 20120217
helper_scripts.sh	2013-06-08	jbork59	[r32] 20130607
index.html	2013-06-08	jbork59	[r31] 20130607
start.console.sh	2012-02-26	jbork59	[r27] 20120226
start.symposia.sh	2013-06-04	jbork59	[r30] 20130603
symposia-console.cpp	2012-03-02	jbork59	[r29] 20120301
symposia.cpp	2013-08-27	jbork59	[r34] 20130826
symposia.h	2012-02-26	jbork59	[r27] 20120226
symposia.php	2011-11-12	jbork59	[r2] 20111111 initial checkin

Figure 3: Subversion Repository of Symposia Source Code Files

Months later I added the symposia project to Sourceforge, and began maintaining two source code lines, one locally and one on Sourceforge. It was not until the formal idea of working code was established that I consistently commented and checked in code revisions of this and my other projects. Now the archive of source code revisions is permanently established and accessible to browsers through the hosting website (Figure 3), and the scope of the changes made in the region of the aforementioned comment can be discerned by the symbols and coloring added by the Subversion web interface (Figure 4).



```
@@ -828,7 +886,10 @@
                }
                else
                        espeak_error = espeak_SetParameter(espeakRANGE, DEFAULT_RANGE, 0);
-
+       }
+
+       if(!just_initialized)
+       {
                memset(&voice_select,0,sizeof(voice_select));
                if(voice)
                {
@@ -838,11 +899,28 @@
                }
                else
                        voice_select.name = NULL;
-
-               if(language)
-               {
-                       voice_select.languages = language;
-                       fprintf(output, "ensound: language [%s]\n", language);
+       }
+
+       if(just_initialized)
+       {
+               /* 20120227 add variant to language specification (always male voice) */
+               if(language || variant || age)
+               {
+                       if(language)
+                               voice_select.languages = language;
+                       if(variant)
+                               voice_select.variant = variant;
+                       if(age)
+                               voice_select.age = age;
+                       fprintf(output, "ensound: language [%s] gender [1] variant [%d] age [%d]\n", l
                }
                else
                        voice_select.languages = DEFAULT_LANGUAGE;
@@ -852,12 +922,9 @@

                close(2);
        }
-       else
-       {
-               fprintf(output, "ensound: espeak already initialized; sample_rate [%d]\n", sample_rate
-       }
```

Figure 4: Changes Made in Revision 28 to symposia.cpp

Documenting this one change in itself is not significant. It is the long-term practice of paying attention to the evolution of the source code that suggests a speculative reimagining of where philosophical insights may arise: not only, not exclusively through textual discoveries of the sort traditionally found by following citations, footnotes, and bibliographic trails, or even following an initial passage of algorithmic, distant reading; not only, not exclusively through accidental outcomes of deformative experiments, either. Adding parameters to enable each speaker to exhibit geno-song characteristics, however crude they are initially synthesized, responds to arguments made by a number of theorists in audio texts and technology discourse networks though an act of philosophical carpentry that can be built, tested, evaluated, and further modified, rather than providing additional textual commentary as the form of criticism.

This project was developed in the context of the coursework in audio texts and technology, which featured Jonathan Sterne's *Audible Past*, which argues that sound technologies are social artifacts all the way down. One of its tasks was to respond to this author's apparent aversion to the synthetic as he develops a history of training in listening technologies, from the invention of the stethoscope to Edison's faint, scratchy phonograph recordings to modern high-fidelity stereo. Thus when he ponders “the aural dimensions of virtual reality, we need to consider audio engineers' century-long obsession with creating what we would now call virtual acoustic spaces in recordings” (338) rather than researching speech synthesis technologies, where he would come across formant synthesis. Perhaps this is because Sterne finds recorded sound itself an exteriority, a resonant tomb lacking the interior self-awareness of conversational partners (290). If the dominant paradigm of collecting texts and artifacts led to a disregard for voice in its original form in favor of forms suitable for performing social functions, and performance has been transformed in order to be reproduced—think Kittler mocking Lacan thanking his microphone—then synthesized voice is truly postmodern simulacra, a simulation of a model. Consequently, Sterne concludes “there are likely as many unexpected connections in the audible present as there were in the audible past. But our speculation should be aided by research, for reality is often stranger and more fascinating than anything we can make up” (338). Using this critical programming exercise, I beg to differ.

Ensoniment

In the study of texts and technology, Plato's *Phaedrus* is often invoked as an early critical work on the efficacy of writing as a knowledge management technology. Neel suggests it is an intentionally self reflexive evaluation of writing by writing, accomplished by presenting an impossible auditory phenomenon. “*Phaedrus* the written text defines thinking by replacing thinking with itself. Thus when *Phaedrus* gives the rules for itself, without ever doing so directly, it has given the rules for thinking. Writing then becomes the absence, or series of absences, that allows itself to be filled up with thinking, for without writing there is no thinking of the sort embodied in *Phaedrus*, which absolutely never could happen as conversation. Nobody talks or ever has talked like the speakers in *Phaedrus*. Plato's attack on writing implies some position outside writing where writing itself can be seen whole, evaluated, and regulated” (37). This position outside writing seems even more evident in the *Symposium*, which proposes an equally unlikely auditory phenomenon.

Near the beginning of Plato's *Symposium*, the participants agree to dismiss the flute players and to avoid excessive intoxication in order to deliver a series of speeches on the topic of love. Is Plato speaking of the seductions of writing as a form of *pro*gramming? With Derrida argue the Greeks were aware of writing as a sort of language machine, a mechanical method in the same category as geometry and dice, and with the power of a drug (*pharmakon*). This turn of the plot could be interpreted as a rhetorical operation acknowledging the requirements of the available media, for no sound recording or reproduction technologies existed in Plato's day that could have captured the event apart from a written transcript. Yet more than other Platonic texts, the *Symposium* resembles a dramatic production that could be recast today in a cinematic or multimedia, virtual reality environment. The latter option opens the hitherto visual world of classical texts to possibilities of *ensoniment*, employing Sterne's term for a disciplined mode of listening that had to be invented.

What other insights were reached by developing this software project? The overall idea of revisiting the corpus of ancient textuality with not just probes of algorithmic criticism, or making it the subject of another digital archive, but instead as content to feed reading machines that give voice to something that has been silently read in translation, is of course the primary outcome. Secondary orality as proposed by Ong and McLuhan was acknowledged to be based on advanced, primary print literacy, and was expected to exhibit structural echoes of its disciplined, algorithmic origins—for this very reason it received early on the postmodern label of 'simulacrum' and now the more nuanced label 'programmed vision'. Following McGann's suggestion, theory at this stage must follow practice as much as guide it. We don't know what we might discover we already knew from long visits to virtual realities formed from Greek or Latin audioscapes, nor do we know what we might discover for the first time as our reading, listening, speaking, writing, and thinking practices change through long acculturation with these virtual worlds.

The sheer struggle to create the original, optimistic implementation that I specified at the outset—an immersive environment similar to the first-person shooter games I had played—taught me as a humanities theorist what I already knew as a seasoned commercial software developer: an enormous amount of effort is required to build something from scratch. In fact, the prototype version of `symposia` that I have presented publicly eschewed the virtual environment altogether, and instead relied on a physical simulation of a virtual environment. Even placing three or four laptops in a room as if they were audio sources arrayed in a three-dimensional virtual environment and getting them to each speak their part at the proper time was a challenge. In this case, however, it was a challenge of manageable complexity, and not overwhelming like the full-blown, original specification, or altogether impractical under the 1990s conditions when I first pondered it.

Tapoc Remediates Literacy

Continuing to track the orality, literacy, electracy progression, the `tapoc` project is the point at which the dissertation reflexively and recursively feeds back into itself, and ceases not to write itself, or simply begins to write itself; it is automated textual production. Thus the acronym "toward a philosophy of computing" functions as the title of the dissertation, the software project, and a means to do what it says—working code as a way to philosophize. Like `symposia`, it has concrete outcomes that enable philosophically minded technologists to extend the code base to produce new experiments and places for contemplation. Its revision history also illustrates an ambitious preliminary objective that turned out to be much more difficult to implement, yielding a hybrid outcome that helps clarify roles in cyberneticly inspired human machine symbiosis. I say cyberneticly inspired because of my reflexive role as the programmer working behind the scenes to produce my own doubly mediated, programmed phenomenal reading and writing experiences—a manufactured place for thinking through literacy to transpire—where I became aware of the situated context of my reading, my rereading, of notes I made to myself days, months, and years before on putatively similar topics, with similar intent to philosophize about computing.

Like `symposia`, `tapoc` relies on a surrounding infrastructure of already working and easy to interface components, such as the MySQL relational database engine, the Apache webserver, and the Perl CGI tools for creating dynamic HTML content from HTTP requests to the webserver by web browser clients, in this case Firefox, which is its primary user interface. This platform has been referred to as LAMP (Linux+Apache+MySQL+Perl/PHP), and for at least a decade has been widely used for rapid development of complete web-based technologies and for providing a comprehensive user interface. Table 4 presents the primary components of `tapoc` in terms of the custom code I wrote that eventually made it into the Subversion repository described below. As in the `symposia` project, there are certain sections of various source files that were sites of many revisions. These became key working code places, and can be found by examining source files for comments, as well as viewing the version differences via the Sourceforge Subversion interface.

Table 4: Source Code Files for Tapoc

Source File	**Language**	**Functional Description**
Arguments-tapoc.sql	MySQL SQL	Table dump of session arguments for web interface
build-JournalInfo-tables.pl	Perl	Utility script to update table data
build.tapoc.sh	BASH	Utility script to build software modules
gentapoc.cgi	Perl	Web console for manipulating items and notes
gentapoc.php	PHP	Web page to serve up content from home page
index-sourceforge.html	HTML	Localhost/Sourceforge.net project home page
Items-tapoc.sql	MySQL SQL	Table dump to notes files and images
Journal.h	C++	Header file for C++ Journal class and other defaults
journal.cpp	C++	Class implementation for Journal object
journal_cmd.cpp	C++	Command line program to manipulate class object
Makefile_tapoc	GNU Make	Makefile to build project
Notes.sql	MySQL SQL	Table dump of notes objects to build dissertation

The `tapoc` project began as a means I developed to manage personal journals I have kept since the early 1990s. After experiencing the literal loss of memory when a few years of Word Perfect 6.1 files became unreadable due to my forgetting a password I had embedded in them, and the challenge of updating these documents as I switched to other word processor programs and new versions came out, I decided to switch to HTML as a format that might persist for more than a few years, and remain application agnostic. Soon, I had ten years of journals, each year consisting of a relatively large (multiple megabyte) HTML file that consumed a substantial portion of my computer's resources merely to open and display in the browser. In the early 2000s, when LAMP was being hailed as the quickest, easiest technology platform for creating dynamic webserver hosts using all free, open source software, I began configuring a webserver that delivered portions of my journals to me in a browser, rather than an entire year at a time. To make it work I had to impose some editing standards in my journals, beginning each new day's entry with the month, day and year in bold font, and adding a bookmark with the pattern journal_YYYYMMDD, for example `journal_20151011`. By passing different arguments to the webserver in the HTTP request, the software I was iteratively developing would return different views of the concatenated yearly journals, starting with a simple reverse sequence from the most recent entry back a few days or a few months. A view I find the most interesting and useful is what I call the "this day years back," showing the journal entry for the selected month and day of the year in reverse chronological order, along with navigational hyperlinks to make it easy to step back through the years to recall what I was doing and thinking, as well as get a sense of whether this particular day has been popular for journaling or not.

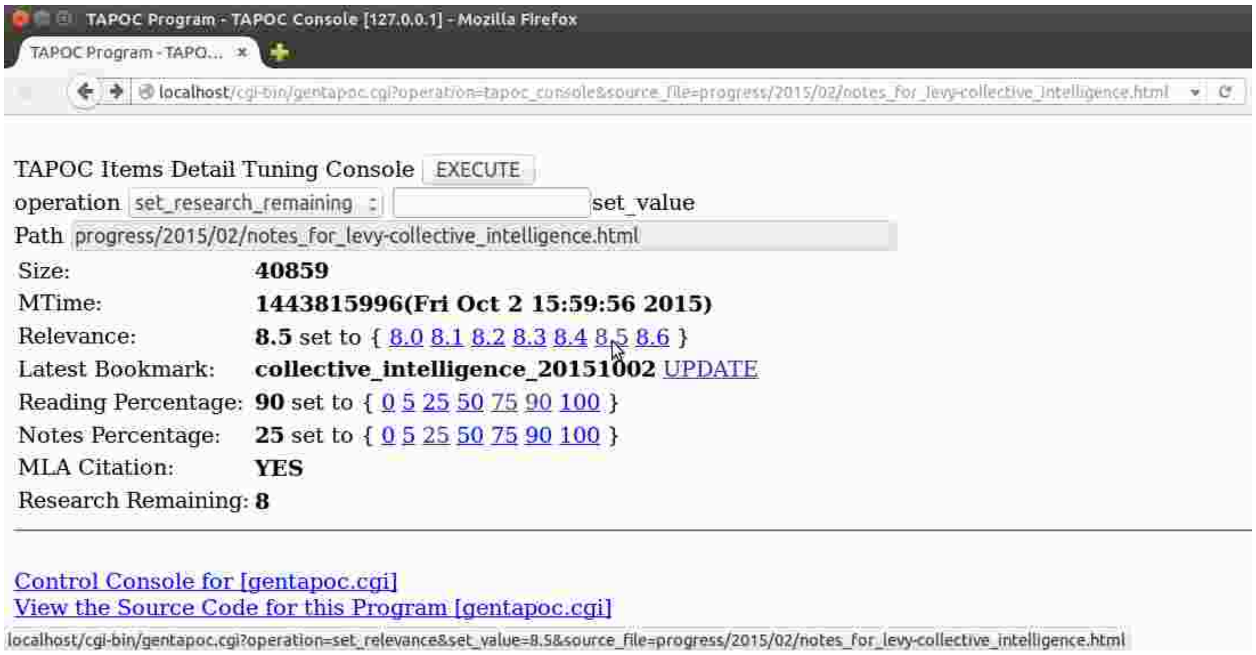
At the heart of the journal software is a C++ class `Journal` that implements an ontology of what I iteratively determined to be the crucial characteristics of my journal entries, and a collection thereof. At the same time, my journal writing habits evolved to better accommodate this design to ensure the entries were parsed correctly. This practice bled over into my note taking for the texts I was reading, both before and after I entered the Texts and Technology doctoral program. For each text I created an HTML notes file consisting of key citations that I underlined while reading, and the notes I wrote in the margin, substantially reconsidered and extended during the focused act of typing in the notes file and rereading the original text in the process. Soon I was including "timestamp bookmarks" indicating when I read the section, so that my traversal through a text could be recapitulated by finding and ordering all of these bookmarks. I had to extend the format to include a single alphabetic letter after the timestamp, since I normally made many notes in one text in one daily reading. Now the hundreds of texts, and thousands of individual bookmarked notes, could be programmatically manipulated.[119]

The thought of writing software that would write my dissertation transitioned from ridiculous to feasible. First, a substantial amount of time was required to assign values to the database contents that were being automatically inserted as new notes files were added from the exam reading lists and new research. This was done by running the Perl script `build-JournalInfo-tables.pl`, which is invoked as part of the GNU `make` process for the tapoc project from its `Makefile`. I already had a table called `Items` that identified all of the HTML personal journals and notes files for the resources that I used in each of my three exam reading lists. Within each notes file were those timestamp bookmarks to each notable citation that I had dutifully typed in over the years of reading. To transition into writing the dissertation, I planned to devise a table of contents with chapters, headings, and subheading levels, and then assign each relevant entry in the `Notes` table to its intended place. Consequently, I was forced to come up with an overall outline of the work, laying out its major argumentative path, and then plan to fill in the details by selecting key passages from the readings and embellishing their accompanying notes.

To get a rough idea of which texts belonged to which chapters, and measure the extent of the remaining note entry effort to accomplish for those texts, I evaluated each entry in the `Items` table and assigned it to a particular chapter, and subjective measures of reading

119 I prefer the term programmatic to algorithmic to shift focus from highlighted unit operations to the overall implementation producing the output, which consists of thousands if not millions of algorithms distributed throughout the system, plus a lot of unexciting connective tissue that is not algorithmically interesting.

percentage completed and notes entry percentage completed, as well as an estimate in the number of hours remaining to spend with that text. To facilitate this process I enhanced the Perl program `gentapoc.cgi` that is accessed via the webserver to provide a 'console' for making these adjustments to the data in the `Items` table when fed the appropriate HTTP request from the browser. Figure 5 shows the `Items` table entry for Levy's *Collective Intelligence* being associated with chapter five.


TAPOC Program - TAPOC Console [127.0.0.1] - Mozilla Firefox
TAPOC Program - TAPO...
localhost/cgi-bin/gentapoc.cgi?operation=tapoc_console&source_file=progress/2015/02/notes_for_levy-collective_intelligence.html

TAPOC Items Detail Tuning Console EXECUTE
operation set_research_remaining set_value
Path progress/2015/02/notes_for_levy-collective_intelligence.html

Size:	**40859**
MTime:	**1443815996(Fri Oct 2 15:59:56 2015)**
Relevance:	**8.5** set to { 8.0 8.1 8.2 8.3 8.4 8.5 8.6 }
Latest Bookmark:	**collective_intelligence_20151002** UPDATE
Reading Percentage:	**90** set to { 0 5 25 50 75 90 100 }
Notes Percentage:	**25** set to { 0 5 25 50 75 90 100 }
MLA Citation:	**YES**
Research Remaining:	**8**

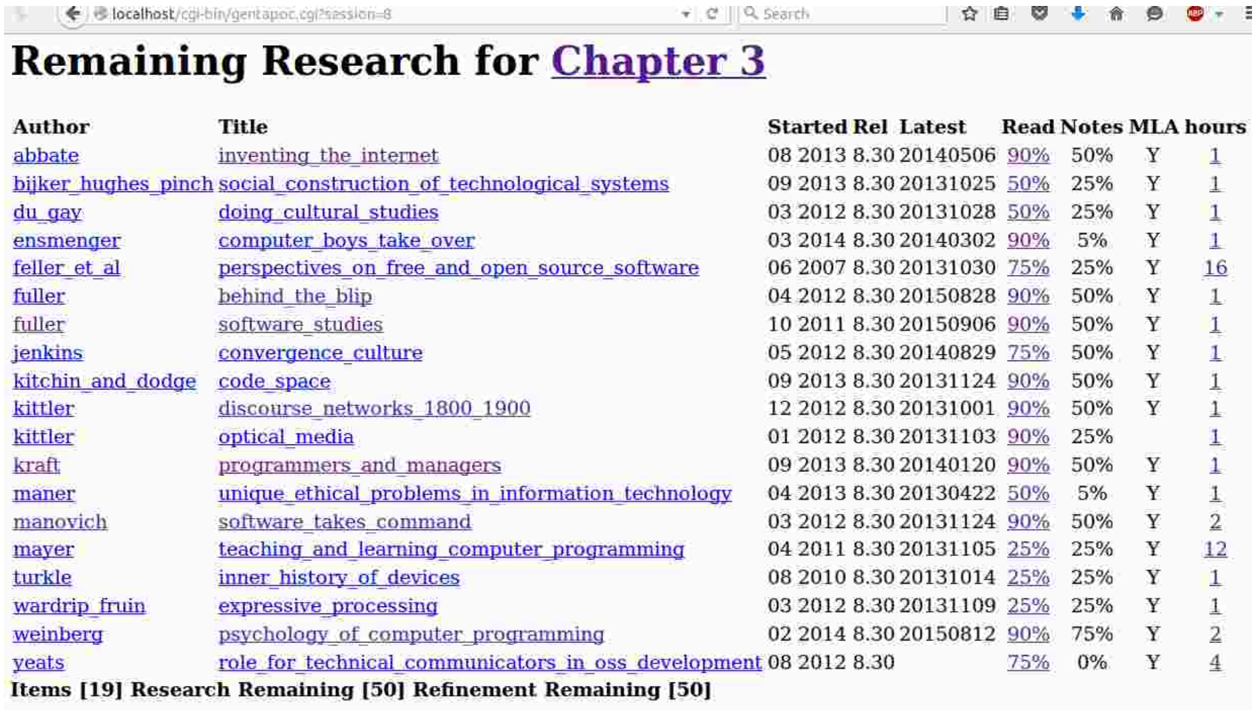
Control Console for [gentapoc.cgi]
View the Source Code for this Program [gentapoc.cgi]
localhost/cgi-bin/gentapoc.cgi?operation=set_relevance&set_value=8.5&source_file=progress/2015/02/notes_for_levy-collective_intelligence.html

Figure 5: Items Detail Tuning Console

As the `Items` table was already well developed from my prior journaling activities and candidacy exam, and populated with hundreds of entries including some that I now wished to include in my dissertation, the `Relevance` field was recycled to designate an item belonging to the dissertation by virtue of a whole number value of 8, and the chapter destination by the tenths decimal place (8.0 for undefined, 8.1 for chapter one, 8.2 for chapter two, and so on).

Creating this utility taught me how the typical LAMP website manipulates data based on requests encoded in hyperlinks served up on the fly, and how I could use this technique to create my own 'consoles' for modifying database data in a controlled manner appropriate to the plans for the `tapoc` project, versus the free-form editing capabilities of a generic database query tool like MySQL Workstation. It's worth noting that a hard-mastery style programmer would not be satisfied with this hack, and had I insisted on formally designing and engineering this project with the rigor I perform at my day job, it would still be in the design stage today. I had to provide myself with a tool that could be immediately used, while still remaining flexible and extensible for the long-term goal of using it to generate the final dissertation as a bookmarked PDF.

After the hundreds of items that would become my List of References were assigned chapters and degrees of completion, I could produce a schedule view in order to identify and estimate the time required to get all of the reading and note taking completed for a given chapter. The schedule is generated on the fly by the `journal` C++ program (invoked by `gentapoc.cgi`) querying the `Items` database. My intentions overshot my available time, so the research for chapter three was abridged, as Figure 6 shows how much work was left on the table. Paying heed to this schedule forced me to reduce the overall scope of the dissertation dramatically, as chapter four was initially devoted to Philosophical Programmers, and chapter five was the content you are reading now. I subsequently moved that material to chapter six as a placeholder for future endeavors.



Remaining Research for Chapter 3

Author	Title	Started	Rel	Latest	Read	Notes	MLA	hours
abbate	inventing_the_internet	08 2013	8.30	20140506	90%	50%	Y	1
bijker_hughes_pinch	social_construction_of_technological_systems	09 2013	8.30	20131025	50%	25%	Y	1
du_gay	doing_cultural_studies	03 2012	8.30	20131028	50%	25%	Y	1
ensmenger	computer_boys_take_over	03 2014	8.30	20140302	90%	5%	Y	1
feller_et_al	perspectives_on_free_and_open_source_software	06 2007	8.30	20131030	75%	25%	Y	16
fuller	behind_the_blip	04 2012	8.30	20150828	90%	50%	Y	1
fuller	software_studies	10 2011	8.30	20150906	90%	50%	Y	1
jenkins	convergence_culture	05 2012	8.30	20140829	75%	50%	Y	1
kitchin_and_dodge	code_space	09 2013	8.30	20131124	90%	50%	Y	1
kittler	discourse_networks_1800_1900	12 2012	8.30	20131001	90%	50%	Y	1
kittler	optical_media	01 2012	8.30	20131103	90%	25%		1
kraft	programmers_and_managers	09 2013	8.30	20140120	90%	50%	Y	1
maner	unique_ethical_problems_in_information_technology	04 2013	8.30	20130422	50%	5%	Y	1
manovich	software_takes_command	03 2012	8.30	20131124	90%	50%	Y	2
mayer	teaching_and_learning_computer_programming	04 2011	8.30	20131105	25%	25%	Y	12
turkle	inner_history_of_devices	08 2010	8.30	20131014	25%	25%	Y	1
wardrip_fruin	expressive_processing	03 2012	8.30	20131109	25%	25%	Y	1
weinberg	psychology_of_computer_programming	02 2014	8.30	20150812	90%	75%	Y	2
yeats	role_for_technical_communicators_in_oss_development	08 2012	8.30		75%	0%	Y	4

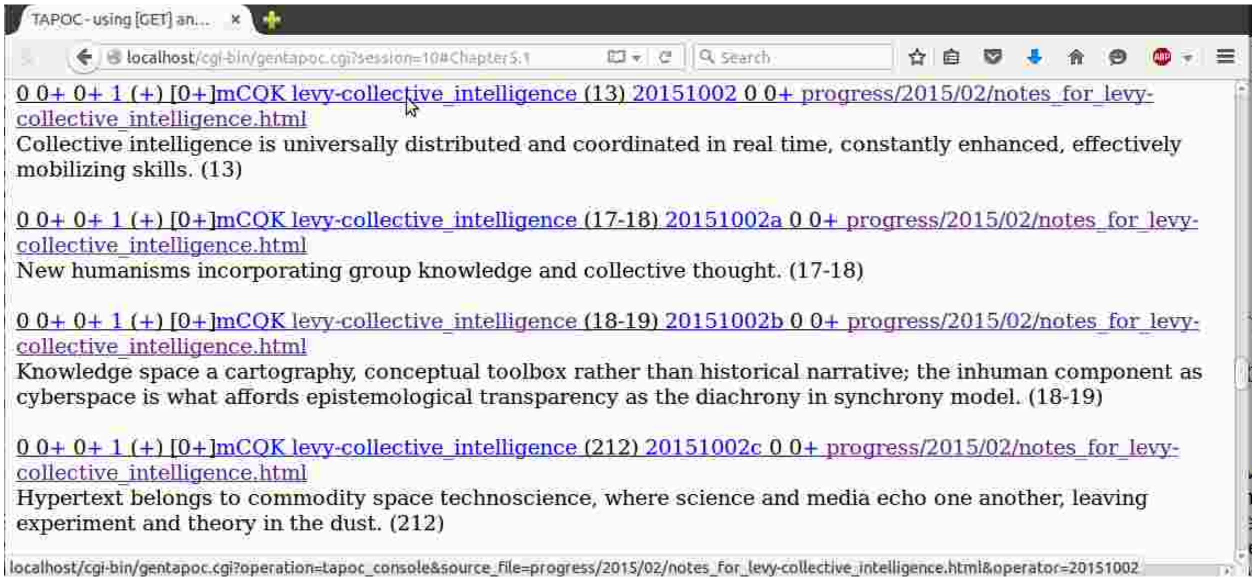
Items [19] Research Remaining [50] Refinement Remaining [50]

Figure 6: Remaining Research for Chapter 3

The essence of the `tapoc` dissertation building process was assigning chapter, heading and subheading designations to each note from each item with remaining research. To facilitate this task that I would spend months performing, I enhanced the `journal` C++ program invoked by `gentapoc.cgi` to produce a view of undefined notes. At its heart is a single database query in `journal.cpp` that produces a result set that is parsed in a `for` loop to produce a web page listing all the `Notes` table entries whose `Chapter` value is `0`, that is, undefined.

```
query = "select Chapter, Heading, SubHeading, InterstitialSequence,
RelevanceLevel, TextName, PositionStart, TimestampBookmarkExtra,
CitationOffset, CitationSentences, Path, Lexia from Notes where Chapter=0 and
(RelevanceLevel=0 or (RelevanceLevel>2 and RelevanceLevel<10)) order by
Heading, SubHeading, TextName, cast((trim(leading '(' from
substring_index(PositionStart, '-', 1))) as unsigned)";
```

These entries are presented as hyperlinks to bring up another console view or assign characteristics directly from the undefined notes page, as shown in Figure 7 for the notes from *Collective Intelligence* I have yet to incorporate into the next chapter.


TAPOC - using [GET] an...
localhost/cgi-bin/gentapoc.cgi?session=10#Chapter5.1
Search

0 0+ 0+ 1 (+) [0+]mCQK levy-collective_intelligence (13) 20151002 0 0+ progress/2015/02/notes_for_levy-collective_intelligence.html
Collective intelligence is universally distributed and coordinated in real time, constantly enhanced, effectively mobilizing skills. (13)

0 0+ 0+ 1 (+) [0+]mCQK levy-collective_intelligence (17-18) 20151002a 0 0+ progress/2015/02/notes_for_levy-collective_intelligence.html
New humanisms incorporating group knowledge and collective thought. (17-18)

0 0+ 0+ 1 (+) [0+]mCQK levy-collective_intelligence (18-19) 20151002b 0 0+ progress/2015/02/notes_for_levy-collective_intelligence.html
Knowledge space a cartography, conceptual toolbox rather than historical narrative; the inhuman component as cyberspace is what affords epistemological transparency as the diachrony in synchrony model. (18-19)

0 0+ 0+ 1 (+) [0+]mCQK levy-collective_intelligence (212) 20151002c 0 0+ progress/2015/02/notes_for_levy-collective_intelligence.html
Hypertext belongs to commodity space technoscience, where science and media echo one another, leaving experiment and theory in the dust. (212)

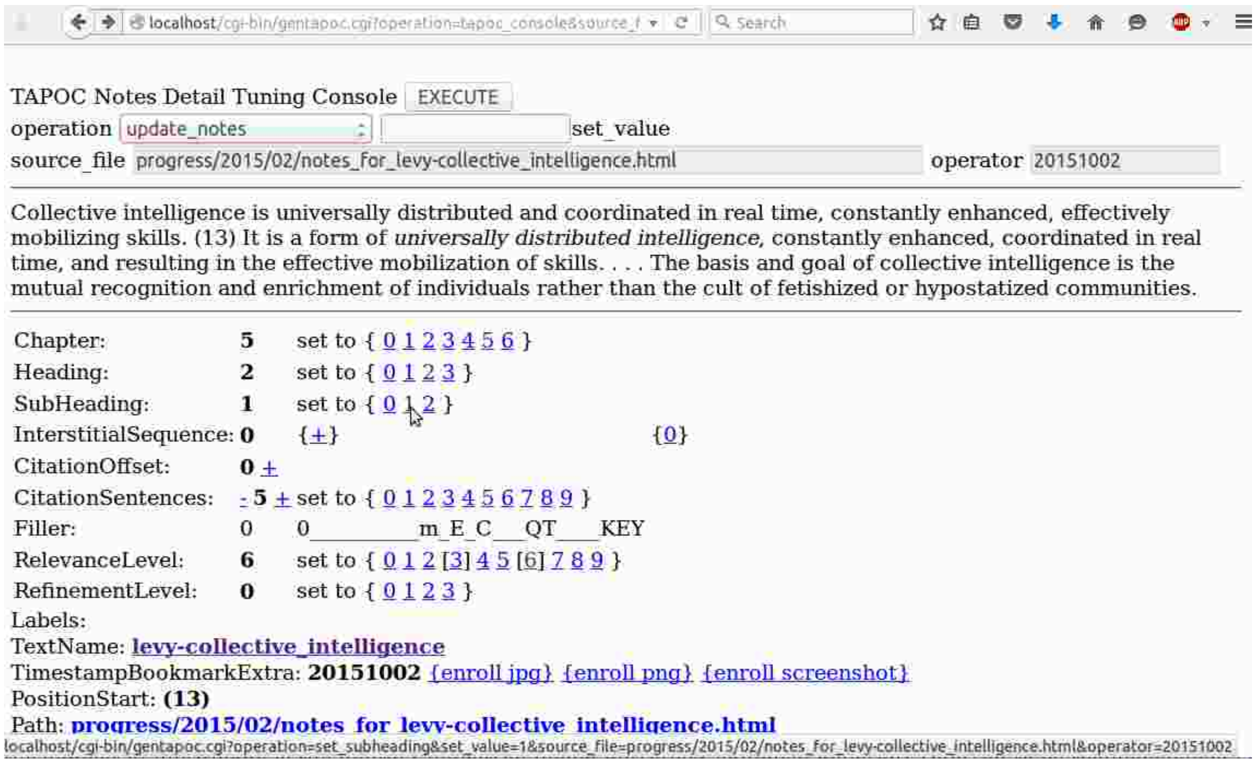
localhost/cgi-bin/gentapoc.cgi?operation=tapoc_console&source_file=progress/2015/02/notes_for_levy-collective_intelligence.html&operator=20151002

Figure 7: Undefined Notes for Collective Intelligence

Through a process I dubbed 'dissertation cow clicking' in honor of Bogost's *Cow Clicker* social media game, thousands of notes were organized into the section I felt they would be useful for writing the dissertation. Each note derived from an HTML rendering of the marginalia and new notes I had entered, along with a number of sentences of text cited from the source. Other entries were derived from my yearly HTML journals, consisting of subsets of daily entries that I encountered during the “years back” reading method, and new thoughts entered on that day of the current year.[120] Figure 8 shows the console view of the note from page 13 of *Collective Intelligence* after setting the `Chapter`, `Heading`, `Subheading`, number of

120 The span of these citations are controlled by the `CitationOffset` and `CitationSentences` fields in their `Notes` entry to define how they are parsed from the source file.

`CitationSentences` to display, and the `RelevanceLevel` for the entry in the Notes table whose unique primary key is identified by `TextName='levy-collective_intelligence' and TimstampBookmarkExtra='20151002'`. At this point, the entry is what I call *unsequenced*. It is relevant to section 5.2.1 but does not have a specific position in the sequence of sentences that will eventually make up the text of the dissertation—at least that was the vision I had for how the software would write the dissertation for me.



localhost/cgi-bin/gentapoc.cgi?operation=tapoc_console&source_f

TAPOC Notes Detail Tuning Console EXECUTE

operation update_notes set_value

source_file progress/2015/02/notes_for_levy-collective_intelligence.html operator 20151002

Collective intelligence is universally distributed and coordinated in real time, constantly enhanced, effectively mobilizing skills. (13) It is a form of *universally distributed intelligence*, constantly enhanced, coordinated in real time, and resulting in the effective mobilization of skills. . . . The basis and goal of collective intelligence is the mutual recognition and enrichment of individuals rather than the cult of fetishized or hypostatized communities.

Chapter:	**5**	set to { 0 1 2 3 4 5 6 }	
Heading:	**2**	set to { 0 1 2 3 }	
SubHeading:	**1**	set to { 0 1 2 }	
InterstitialSequence:	**0**	{+}	{0}
CitationOffset:	**0** +		
CitationSentences:	- **5** +	set to { 0 1 2 3 4 5 6 7 8 9 }	
Filler:	0	0_________m_E_C__QT___KEY	
RelevanceLevel:	**6**	set to { 0 1 2 [3] 4 5 [6] 7 8 9 }	
RefinementLevel:	**0**	set to { 0 1 2 3 }	

Labels:

TextName: **levy-collective_intelligence**

TimestampBookmarkExtra: **20151002** {enroll jpg} {enroll png} {enroll screenshot}

PositionStart: **(13)**

Path: **progress/2015/02/notes_for_levy-collective_intelligence.html**

localhost/cgi-bin/gentapoc.cgi?operation=set_subheading&set_value=1&source_file=progress/2015/02/notes_for_levy-collective_intelligence.html&operator=20151002

Figure 8: Dissertation Cow Clicking

Going through this process of sorting notes by their likely placement within sections of the dissertation and setting the span of potential citations led to thinking about the role each note ought to play, so I added another field to the table. I intended a `RelevanceLevel` of 6 or higher to mean *quotable*, i.e., my notes followed by the `CitationSentences` worth of text displayed in double quotes with the `PositionStart` identifying the page number, whereas those with a `RelevanceLevel` of 4 to mean *citable*, i.e., to display my notes depending on the text but just citing the page number. A `RelevanceLevel` of 3 stands for footnote or endnote material, and less than 3 for personal notes.

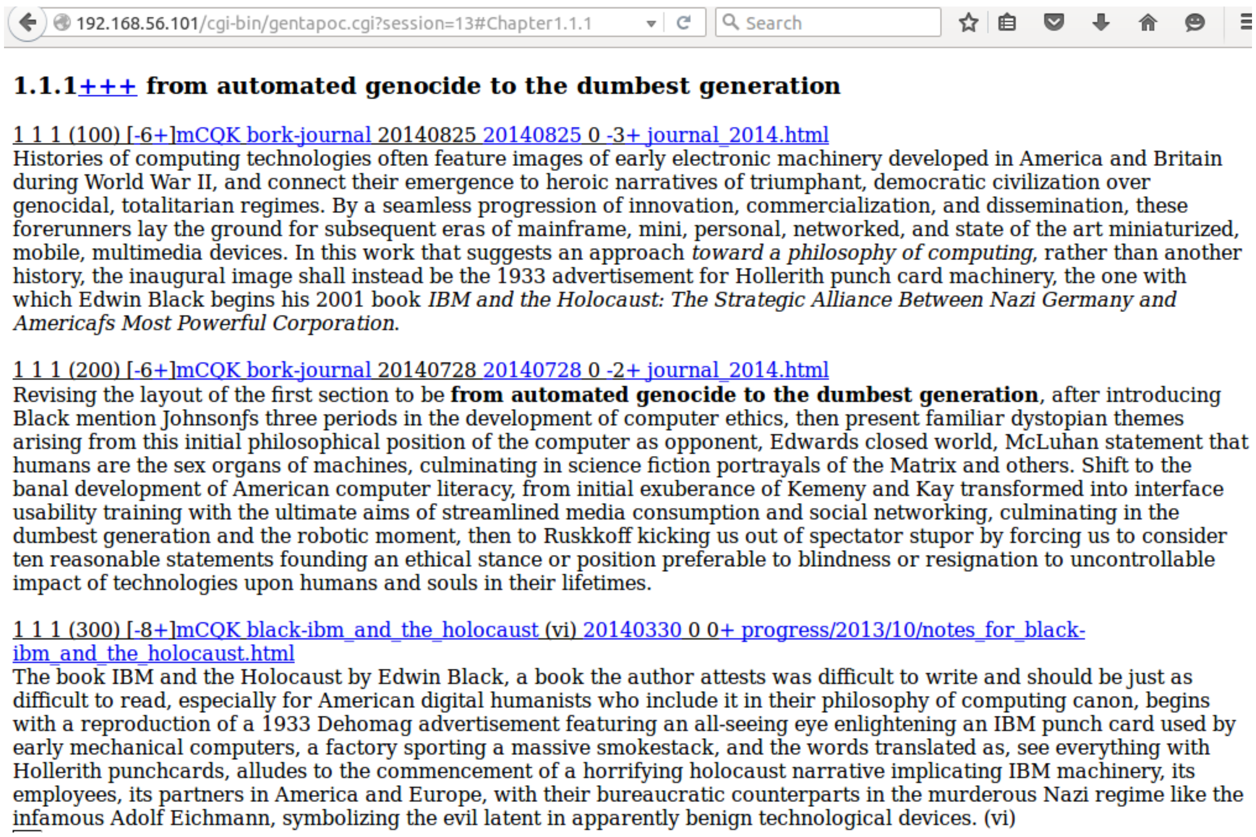
First I needed a view of the unsequenced content by section so I could begin selecting which notes really warranted inclusion in the dissertation text from each theorist studied. Again, `journal.cpp` was modified to produce an new web page view of the notes, ordered by `Chapter`, `Heading`, `Subheading`, and within each subheading, by `TextName`, which is a concatenation of author and title, and finally `PositionStart`, the page number of the citation. The C++ code for this view became the primary work area for thinking about the strings of argumentation forming my thesis, and the block of code in the class function `Journal::display()` where variations of a single database query generated the desired result set from which the web page was built became a site of frequent revision.

The crucial step for turning a subset of these notes, which numbered in the hundreds and soon thousands as I continued to introduce them into the `Notes` table, into a readable text was to begin sequencing them, stringing out the arguments I wished to make while firmly grounding my points with citations from my list of references. After sequencing the ten or so sections of Chapter 1, some of which were subsequently merged and pared down, I had over five hundred blocks of text that could be presented in a form resembling the intended final product. Once again a flurry of working code in the `Journal::display()` function occurred to take another set of arguments from an HTTP query, passed through `gentapoc.cgi`, to call the C++ program `journal`. The ordering was performed by the MySQL database engine based on the revised query, now run in nested `for` loops to produce the entire dissertation or just a given Chapter and Heading, since the resulting web pages were becoming large enough to bring Firefox and my operating system to a crawl when refreshed.

```
query = "select Chapter, Heading, SubHeading, InterstitialSequence,
RelevanceLevel, TextName, PositionStart, TimestampBookmarkExtra,
CitationOffset, CitationSentences, Path, Lexia, if((select count(*) from
Items where Path like
concat('%/',Notes.TextName,'_',Notes.TimestampBookmarkExtra,'.jpg') and
Relevance=1),'Y','N'), if((select count(*) from Items where Path like
concat('%/',Notes.TextName,'_',Notes.TimestampBookmarkExtra,'.png') and
Relevance=1),'Y','N'), if((select count(*) from Items where Path like
concat('%/screenshot_',Notes.TimestampBookmarkExtra,'.png') and
Relevance=1),'Y','N'), if((select count(*) from Items where Path like
concat('%/vpf_',Notes.TimestampBookmarkExtra,'.jpg') and
Relevance=1),'Y','N') from Notes where Chapter=" + Chapter.str() + " and
Heading=" + Heading.str() + " and InterstitialSequence>0 order by Heading,
SubHeading, InterstitialSequence, TextName ";
```

The query also contained subqueries to determine whether there were any images associated with a particular `Notes` entry, as many notes from my personal journals included screen shots or even digital photographs, sometimes of a large whiteboard I used to sketch the relationships between ideas, and I wished all of this content to be automatically generated as part of the viewable output.

Figure 9 shows the beginning of the first section of Chapter 1 with the surrounding hyperlinks to invoke the aforementioned console tool to adjust the entries. A small change to the display routine to suppress these 'tuning' options and metadata would yield a readable text, or so I thought.



1.1.1+++ from automated genocide to the dumbest generation

1 1 1 (100) [-6+]mCQK bork-journal 20140825 20140825 0 -3+ journal_2014.html
Histories of computing technologies often feature images of early electronic machinery developed in America and Britain during World War II, and connect their emergence to heroic narratives of triumphant, democratic civilization over genocidal, totalitarian regimes. By a seamless progression of innovation, commercialization, and dissemination, these forerunners lay the ground for subsequent eras of mainframe, mini, personal, networked, and state of the art miniaturized, mobile, multimedia devices. In this work that suggests an approach *toward a philosophy of computing*, rather than another history, the inaugural image shall instead be the 1933 advertisement for Hollerith punch card machinery, the one with which Edwin Black begins his 2001 book *IBM and the Holocaust: The Strategic Alliance Between Nazi Germany and Americafs Most Powerful Corporation*.

1 1 1 (200) [-6+]mCQK bork-journal 20140728 20140728 0 -2+ journal_2014.html
Revising the layout of the first section to be **from automated genocide to the dumbest generation**, after introducing Black mention Johnsonfs three periods in the development of computer ethics, then present familiar dystopian themes arising from this initial philosophical position of the computer as opponent, Edwards closed world, McLuhan statement that humans are the sex organs of machines, culminating in science fiction portrayals of the Matrix and others. Shift to the banal development of American computer literacy, from initial exuberance of Kemeny and Kay transformed into interface usability training with the ultimate aims of streamlined media consumption and social networking, culminating in the dumbest generation and the robotic moment, then to Ruskkoff kicking us out of spectator stupor by forcing us to consider ten reasonable statements founding an ethical stance or position preferable to blindness or resignation to uncontrollable impact of technologies upon humans and souls in their lifetimes.

1 1 1 (300) [-8+]mCQK black-ibm_and_the_holocaust (vi) 20140330 0 0+ progress/2013/10/notes_for_black-ibm_and_the_holocaust.html
The book IBM and the Holocaust by Edwin Black, a book the author attests was difficult to write and should be just as difficult to read, especially for American digital humanists who include it in their philosophy of computing canon, begins with a reproduction of a 1933 Dehomag advertisement featuring an all-seeing eye enlightening an IBM punch card used by early mechanical computers, a factory sporting a massive smokestack, and the words translated as, see everything with Hollerith punchcards, alludes to the commencement of a horrifying holocaust narrative implicating IBM machinery, its employees, its partners in America and Europe, with their bureaucratic counterparts in the murderous Nazi regime like the infamous Adolf Eichmann, symbolizing the evil latent in apparently benign technological devices. (vi)

Figure 9: Sequenced Output for the First Section of Chapter 1

My elation at the early success of generating the first chapter of this dissertation through my own custom code was quickly supplanted by concern and eventually dread as I began the necessary editing process of the source texts from which the `Notes` objects derived in order to smooth out the flow of sentences based on their new sequencing. Moreover, if the sequencing changed or I added new material, the surrounding notes had to be modified again. This process proved so time consuming that I decided to call this approach a failure and revise the plan. Not that I would abandon the collecting and sequencing processes for the remaining chapters, but rather that the writing of the dissertation text would happen in a word processor, with a lot of copy and paste operations from the web pages produced by `gentapoc.cgi`.

As I commenced writing the first chapter in LibreOffice Writer, I maintained the discipline of keeping the content and sequencing of the `Notes` entries in sync with the free form text being written. Thus the cybernetic paradigm of automated writing shifted from a programmer- and machine-driven to more traditional literary form, yet I still feel that the months spent working the code in conjunction with organizing my thoughts by using that same code I had just modified had a profound impact on the progression of the work as it had been laid out in the proposal. Moreover, I now had a means to gauge the amount of time it would take to complete the other chapters (see Table 5) if I adopted the new method of sequencing each section using the vast "work in progress" alphabetical listing views, and then using the sequenced output as shown in Figure 9 to feed the LibreOffice word processor document.

Table 5: Counts of Items and Notes by Chapter

	Undefined	**Ch 1**	**Ch 2**	**Ch 3**	**Ch 4**	**Ch 5**	**Ch 6**	**Total**
Items (References)	13	14	47	141	9	6	37	267
Unsequenced Notes	498	97	617	3129	694	720	745	6500
Sequenced Notes	0	519	675	1492	115	54	50	2905

This initial setback and the resulting rethinking of the software project supports McGann's contention that failure is an acceptable outcome for digital humanities experiments. He calls them the rewards of failure "because our computer tools are models of what we imagine we know—they're built to our specifications—when they show us what they know they are reporting ourselves back to us" (143). In the case of his experiments with devising suitable markup codes for a digital archive, he and his associates came to understand "that all texts are marked texts," and "the OCR experiments showed that textual ambivalence can be located and revealed at graphical, presemantic levels. . . . For it is a commonplace in both the SGML/TEI and the hermeneutic communities that these codes do not signify in the way that semantic codes do" (145). Likewise, I learned that the seemingly peripheral task for scholarly writing of producing smoothly flowing text by adding the appropriate connective and transitional grammatical elements to discrete units of preformed content can, in fact, take as much time as the initial writing.[121] This realization follows McGann's third lesson: "as the experiments strongly suggested that while every text possesses, as it were, a self-parsing markup, the reading of that markup can only be executed by another parsing argent. That is to say, there can be no such thing as an 'unread' text" (145-146). The smoothly flowing arrangement that I interpreted from the marked-up output of Figure 9 was just that—an interpretation by another parsing agent, namely my imagination. The dissertation was not immanent in the database table as an unread text. Yet nor was it simply an ordered hierarchy of complex objects (OHCO). The working code had to make it such, and since I was using the powerful C++ Standard Template Library (STL) to perform dynamic manipulations that literally created and morphed these complex objects on the fly, I felt it transcended this conception.

How does `tapoc` remediate literacy? Beyond just revealing markup codes—which used to be a function of commercial word processors—working code *places* are revealed. It responds to the double mediation performed by digital media that led Kittler to apologize to his readers for

121 The same outcome occurs in software development projects. Novice software engineers frequently underestimate the time it will take them to complete a coding assignment based on the description of the problem and some preliminary code work. The unexciting connective tissue, error handling routines, and documentation are the programming counterparts to these manual adjustments to my sequenced `Notes` entries to produce a smoothly flowing text.

writing *under* Microsoft, within the constraints of protected mode microprocessing, and wholly at the mercy of software and electronic circuits to retrieve and render it from its native form of extreme inscription. Under `tapoc` the writing emerges from machine processes that were at least partially directed by me the writer functioning also as me the programmer. I know how the sequence of sentences and larger ordering of paragraphs and sections came to be through the execution of iterations of database queries, C++ markup parsing routines, and the well documented passage through the LAMP architecture that offers up everyday web browsing that we take as our customary reading and writing environment.

Moreover, this form of literacy rises above the absent-minded 'cow clicking' that Bogost and Heim accuse post-postmodern network dividual cyborgs of perpetrating. There, thought-provoking pauses while options are considered and unknown knows are resolved by testing solutions get shunted by the immediate response and continuation of the programmed visions presented by user interfaces designed to capture and hold attention. By cycling back and forth between composition and coding, the literary product, including this text you are reading, but also the sundry, ephemeral, related views created by different parameters fed into `gentapoc.cgi`, bears the structural imprint of the long revision history of the software system shot through with my evolving thoughts about the philosophy of computing.

Fossification

Kittler claims speech became immortal through its passage into technological reproduction, epitomized by Lacan thanking the thoughtless microphones to which he delivered his lectures, yet he elsewhere acknowledges that writing now takes place at the microscopic level of extreme inscription and the double mediation performed on our behalf by computing machinery that constitutes an essential component of our extended minds. This entails the present or future risk that the machines will take that data off into places impossible for us to follow them. Consequently, the permanence of speech and writing committed to technological media is assumed but not guaranteed. A second articulation of this thought is made by Marilyn Deegan with regards to the components of digital scholarly archives, and the appropriate tasks for digital editors, implying that certain components of digital texts remain eternally interesting and worth preserving, whereas others, namely the software and encoding formats that provide the interface to the readers, ought to be considered ephemeral. She proposes that “we might expand the distinction to a fivefold one: data, metadata, links, program, and interface. The first three contain the intellectual capital in an edition; the last two are (should be?) external. However important the programs used to create and deliver the edition and however important the interface through which it is accessed, scholars must always remember that these parts of any electronic edition are the least durable” (*Electronic Textual Editing* 365-366).

This distinction maintains an assumed separation of content and technological media that remains plausible for symbolic texts consisting of natural, mother tongue human languages, but becomes murky as digitally native electronic literature begins to enter the range of material deemed worthy of scholarly study, for these works are more formally bound to the various cultural software applications and their idiosyncratic composing practices with which they are produced, transmitted, stored, reproduced, and remixed. Anybody who has lost contact with their prior work because its only traces are on diskettes encoded in obsolete word processor file formats knows that truth of this. Emulation and virtual machines provide some protection against this form of bit rot, and the massive storage capabilities of the present day reduce virtual machines containing an entire life's work to tiny units that are true Alephs when accessed.

Yet one force continues to overpower even the most advanced consumer technologies, holding this potential at bay: United States and international copyright laws. Indeed, according to Lawrence Lessig and many others, code has become law because copyright laws have been implemented in code to govern our interaction with others' intellectual property. Digital rights management and access systems regulate behavior by controlling it before the act of lawful or unlawful use, by creating and managing the code/space in which we interact with such intellectual property. Moreover, the grip of code is strengthened by legislation like the DMCA to discourage and penalize the investigation of these software components that regulate our access to phenomena.

In the most extreme situations, where every request is associated with a particular individual and logged, the panopticon of pervasive surveillance Foucault envisioned has been realized. Its everyday form is the asymmetric relationship Jaron Lanier presents between the multitude of users and an oligopoly of siren servers that orchestrate the majority of cyberspace encounters. In an extraordinarily banal Faustian bargain, these users give up their own intellectual property to the siren servers in exchange for the illusion of free access to social media networks, games, news, entertainment, porn, and so on. Because that intellectual property takes the form of mouse clicks, short messages, video clips, invisible browser cookies, and other unit operations, it does not seem substantial, like an original short story or dramatic performance. Yet the sum of those blips becomes our cyberspace legacy, concretized in data trails, browsing histories, and other aggregations now belonging to the siren servers as their intellectual property to process, trade among themselves, and feed back to us in the form of suggested stories, advertisements, and future programmed visions in the years to come.

The situation could have turned out much worse than it has, in terms of the distribution of freedom between users and producers if free, open source software had not come into prominence. Richard Stallman was prescient in developing the notion of Copyleft and working with the Free Software Foundation to turn it into actionable legal verbiage as the GNU General Public License (GPL). Software developed and licensed under the GPL undergirds much of the Internet infrastructure, including the siren servers Lanier critiques, in addition to the operating systems of personal devices used to access content. Many of us remember the time before the wide proliferation of foss, when it seemed like Microsoft would control computer workstations, programming languages and the Internet itself, were it not for protracted legal battles over its purported monopoly. Those concerns have diminished in the present heterogeneous computing ecology, only to be replaced by concerns over the disposition of the content distributed to and exchanged among users of these platforms. Creative Commons licenses for creative works have been developed that parallel many aspects of the GPL Copyleft, yet a gap remains between the mechanisms that serve up content and the content itself, maintaining their ontological distinction for the sake of capitalism in spite of their material coextensiveness as protocological phenomena.

It occurred to me after enacting it in code for many years while developing first my personal journal, and then the `tapoc` project, that a response to this problem is to *embed the content into the code itself*. That is, rather than publish the content under a CC license or under the FDL (Free Documentation License) accompanying the GPL source code, make it part of the source code, so that it is once and for all released and given over to circulate freely in the machinic—by virtue of Stallman's four freedoms—as well as human collective intelligence. I call this process *fossification*, a play on the term 'ossification' that I read in Tauel Harper's piece “Smash the Strata,” referring to the perennial problem of power becoming entrenched and eventually immobilized in political systems. Harper, a proponent of what I alluded to in the introduction may devolve into foss hopes, believes “new forms of technology such as the Internet and virtual reality might facilitate a rhizomatic reterritorialization of the incumbent Urstaat. . . . More simply put, the fact that technology is continually developed by capital suggests that technology inherently deterritorializes and resists the ossification that otherwise poses a problem for political systems” (126). Other critics like Boltanski and Chiapello and Lessig have pointed out that capitalism does an excellent job deterritorializing ossified structures like copyright law enacted in the print era and reterritorializing it to extend into software and data streams, so the now mythic free-for-all of the early Internet seems increasingly striated by these siren servers that are considered the darlings of the projective city.

Fossification innoculates literary work from the territorializing efforts of capitalism by securing it into the perpetual free space of GPL program source code. This is crucial for electronic literature that does not adhere to the traditional editorial boundaries of data, metadata, links, program and interface that Deegan enumerates for scholarly editions of print media because, as working code, it exists on all these levels as a diachrony in synchrony. At the same time, I see fossificaiton as a solution beyond the default submergence of meaning to data that Krӓmer describes as a form of digitalized existentialism speaking out from Kittler's texts. “The culminating points within the tradition of literary studies . . . are transformed into the absurd with the binary code that cannot be expressed by humans, and with modern data processing as a textile that cannot be read by human eyes. This 'absurdity' can be understood in the sense of Kierkegaard as a paradox of the historical, which stands in start opposition to human logic and sensibility, or in the sense of Camus, who counter-intuitively lets the world remain mute to human questions" (106).

What texts are candidates for fossification? First, anything that is already in the public domain, from typographic material to recorded media. So I have done with Plato's *Symposium* by encoding its Greek text into the symposia project. Even the recorded speeches of Lacan need fossification to escape copyright and other licensing traps for true immortality and universal access. Thus, for many texts, on the one hand, fossification cannot occur until a future date *when all copyrights expire*. On the other hand, currently unpublished and new content can immediately be passed over into this free space by turning it into program code. Importantly, then, and differently from Creative Common licenses, philosophy subsumes fair use by fossification, if you accept its affordances. Not just what is deemed fair use, but any and all use, is permitted by the four freedoms of free software that Stallman articulates and has enshrined in the GPL.

I call this action *philosophical finesse*. It describes something possible through critical programming that is only awkwardly achieved via traditional methods of citation, and prepares future scholarship for the challenging task of working with all the layers of electronic literature. For a substantial portion of current and future work in the digital humanities involves processes of distant and hybrid reading performed by computer software in conjunction with human interpreters. Without materials already prepared for effortless machine reading, and this means not only setting them into the appropriate markup forms, but also validating intellectual property clearance for every piece involved, a great number of potentially interesting thought experiments

—whose real significance, McGann assures us, only arrives after the fact through poeisis-as-theory and uncovering what we did not know we already knew—will never happen.

Pmrek Remediates Electracy

The previous project may seem like an ideal candidate for electracy, which Ulmer nominates as the successor of literacy. However, as I suggested in the introduction, digital humanists following Ulmer's lead tend to emphasize *traces* over *electrons*,[122] as exemplified in digital media of the sort Hayles has surveyed in many of her books. To take the notion of the extended mind to the limits of current human perceptual and cognitive processes and cross over into the machinic requires a more radical departure than extending her claim that “literature can be understood as a semiotic technology designed to create or more precisely, activate feedback loops that dynamically and recursively unite feelings with ratiocination, body with mind” (*Electronic Literature* 134). How these feedback loops operate in nonhuman systems is left open as Hayles does not spend much time at all discussing the source code of any of the examples of electronic literature; her musings on what feelings/body and ratiocination/mind may be in nonhuman systems trace the same boundaries of fantasy as do those surrounding her postulate that nonhuman intelligences exist in representations of it by Talan Memmott's *Lexia to Perplexia* (122). It is with examples from Memmott's work that she presents broken code, “code that is a creolization of English with computer code, evocative of natural language connotations but not actually executable” (123).

122 Following Kittler we ought to distinguish the electronic from the electric as well, paying heed to the crucial action of diodes and transistors in steering electrons as the basis of all digital computing.

What Hayles does assert is that embodiment mediates between technology and discourse, affecting body use and experience of space and time, especially through the link of incorporation. “Formed by technology at the same time that it creates technology, embodiment mediates between technology and discourse by creating new experiential frameworks that serve as boundary markers for the creation of corresponding discursive systems” (*How We Became Posthuman* 205). Thus we are encouraged to experiment with this cyborg boundary by learning machine skills, suggesting not merely playing videogames, but that cultivating knowledge and skill in electronics—answering Knuth's challenge to analyze every process our computers execute in one second—itself may have transformative, synaptogenetic implications.

Turing, describing machine computation, wrote “the idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer. The human computer is supposed to be following fixed rules; he has no authority to deviate from them in any detail” (52). The accepted method of modeling computational processes has been to sketch them out as flow charts to illustrate their sequences, branches, and cycles that they follow with a regularity and discipline humans attempt to achieve through the most extreme disciplinary regimens, hinting at command and obey military metaphors, with the obligatory asterisk noting that in reality they perform these operations at unimaginably high frequencies over equally vast numbers of units. It is through this computational agency that electracy is differentiated from literacy. As Noah Wardrip-Fruin puts it, digital media processes “are separated from noncomputational media processes by their potential numerousness, repetition, and complexity. . . . It is the computer's ability to carry out processes of significant magnitude (at least in part during the time of audience experience) that enables digital media that create a wide variety of possible experiences, respond to context, evolve over time, and interact with audiences" (9-10). Appreciating the alien temporality of digital processes goes hand in hand with Bogost's alien phenomenology, and it is certainly an appropriate competency for anyone wishing to be a philosopher of computing. Another task for critical programming studies, therefore, is to provide tutor texts for what I call *intuiting machine embodiment*.

Machine cognition, if it exists at all, currently resides in electronic circuitry. Unlike literary traces, whose distinct marks form the aggregates from symbols on upwards to lexia to texts that comply with the OHCO thesis, electrons are individually indistinguishable, too numerous to appear as discrete units like traces, move very quickly, and assemble in very high numbers. Controlling their flow in the microstructures of integrated circuits according to plans and situated actions is the primary objective of programming. It is instructive to consider how modern programming languages like GNU C attach time to computational objects to think about these extremely fast, small, and numerous orders of magnitude. It is interesting that the `man` page of built-in documentation for the C programming language talks about 'broken-down time' as an ontological characteristic of computational entities, thingness, existence in the programming language and run time environment. Typing `man mktime` on the BASH command line gives the help "Broken-down time is stored in the `structure tm` which is defined in `<time.h>` as follows:"

```
struct tm {
      int tm_sec;         /* seconds */
      int tm_min;         /* minutes */
      int tm_hour;        /* hours */
      int tm_mday;        /* day of the month */
      int tm_mon;         /* month */
      int tm_year;        /* year */
      int tm_wday;        /* day of the week */
      int tm_yday;        /* day in the year */
      int tm_isdst;       /* daylight saving time */
};
```

Time represented in this manner epitomizes Kittler's time-axis manipulation because the individual temporal components of an object of this data type can be treated by computational processes whose own temporal orders of magnitude are not dependent on any particular temporal aspect of this object. This is another way of saying that for computational objects, like the Aleph and Barthes' mythical speech, “there is no regular ratio between the volume of the signified and that of the signifier. In language, this ratio is proportionate, it hardly exceeds the word, or at least the concrete unit. In myth, on the contrary, the concept can spread over a very large expanse of signifier” (“Myth Today” 90). Furthermore, with broken-down time programs can measure and reckon temporal durations that are exceedingly short in human terms, where the boundary of perceptibility is around 50 milliseconds, but may fall within variegated, diachronous spans in the same general synchrony of run time.

The Pinball Machine Reverse Engineering Kit (`pmrek`) is a critical programming project that makes a study of broken-down time structures deployed by multiple, concurrent processes to regulate the flow of electrons in discrete, physical acts of reading and writing binary digits. Like `symposia` and `tapoc`, it consists of a substantial amount of custom source code that I developed and used for many years, and in its working code places revised and refined various ideas I have had about computing and the human relationship to the machinic. This project differs from `symposia` and `tapoc`, which generate simulacral audio and visual phenomenal in what Castells calls real virtuality, i.e., cyberspace. Instead, `pmrek` is what Manovich calls grey software, the tightly bound control element of a specific high speed digital process control system—a Bally electronic pinball machine from the late 1970s. The original goal of `pmrek` was to reverse engineer its defective Microprocessor Unit (MPU) and replace it with a mostly passive, home made replacement control board driven by a generic Intel x86 architecture personal computer running a GNU/Linux operating system and custom foss comprising of both a kernel module leveraging the relatively deterministic, high frequency *workqueue* feature of the new (at the time) 2.6 version of the Linux kernel for time-critical, high speed operations, and a non-deterministic, relatively low frequency user-space program for less critically timed operations like tracking the game play and scoring.

From my masters thesis, "reverse engineering creates knowledge through research, observation, and disassembly of a part in a system in order to discern elements of its design, manufacture, and use, often with the goal of producing a substitute" (1). I advise using reverse engineering for pedagogy and for resistance to the tendency to ignore the material reality of machinery and the tendency to rely solely on information represented in patterns of symbols rather than the actual material objects that constitute the thing itself being discussed. This common approach results in an unconscious philosophy that is crippled from the start by allowing itself to be shaped by the default consumer technologies constituting the posthuman cyborgs of the early twenty-first century. Pinball machines are classic, second wave cybernetic systems exemplifying switch matrix mediated closed loop feedback control that imbricates message, signal, and information in broken-down time structures exchanged between the Linux kernel and an ordinary user application, bridging the protected mode barrier that Kittler decries. Hayles claims autopoiesis turns the cybernetic paradigm inside out, so I suggest this elementary study of machine embodiment should be done in preparation for thinking about third wave concepts, rather than skipping directly to them.

Table 6 presents the source code files for the `pmrek` project.

Table 6: Source Code Files for Pmrek

Source File	**Language**	**Functional Description**
analyze_testbed_output.php	PHP	Creates a web page analyzing the game play
pmrek.c	C	Source code for Linux kernel module
pmrek.h	C	Header file
pmrek.mod.c	C	Auto-generated source code for kernel module
start.pmrek.sh	BASH	Script to start the game
strikes_and_spares.cpp	C	Custom functions to play Strikes and Spares
strikes_and_spares.h	C	Header file for Strikes and Spares
testbed.cpp	C	User-space program to play the game
user_pmrek.cpp	C	Utility program to simulate game play

Where the major modules of `tapoc` interact with the Apache webserver responding to HTTP requests from a web browser client seeking various views of the MySQL database table data ultimately destined for human reading, and therefore are not overly concerned with timing besides delivering results before the browser or reader's patience *times out*, the major modules of `pmrek` interact with the physical electronic circuits of a pinball machine and the physical setting of the game play—switches, lights, solenoids, scoring, etc. Working code places in `pmrek` deal with simultaneous processes occurring in multiple temporal orders of magnitude—so called 'real time'—where taking heed of both *deadlines* and what I call *shorttimes* are crucial for the machine to operate correctly.[123] A 'shorttime' is the minimum duration that must transpire between some time-critical operations without exceeding other deadlines, but nonetheless far below the human perceptual threshold. In `pmrek` there are a number of control processes occurring in different temporal orders of magnitude for which deadlines and shorttimes must be managed by the working code in conjunction with the affordances of the platform, each of which reflects an aspect of machine embodiment (Table 7).

Table 7: Time-critical Pmrek Control Processes

Process	**Software Component**	**Frequency**	**Deadline / Shorttime**
Game play	`testbed` application	4 Hz	3 ms / 250 ms
Solenoid control	`pmrek` kernel workqueue	333 Hz	1 μs / 26 ms
Switch matrix row read	`pmrek` kernel workqueue	333 Hz	1 μs / 15 ms
Feature lamp control	`pmrek` kernel workqueue	333 Hz	1 μs / 3 ms
Numeric display control	`pmrek` kernel workqueue	333 Hz	1 μs / 3 ms

Game play by the `testbed` application program is all the human observer perceives, but it is just the top level of a beastiary of subsidiary control processes that ultimately touch the physical hardware circuits of the pinball game itself.

123 The `symposia` system, in contrast to both of these, is merely concerned with the individual speaker processes performing their utterances in the right order. Their synchronization is not dependent upon any timing constraints as each processes waits for the globally incrementing `Position` to reach the threshold of their next entry in the `Lexia` table before commencing.

The other component of `pmrek` is its custom interface board to mate the Intel x86 architecture to the existing Motorola 6800 platform components via the classic 8-bit Industry Standard Architecture (ISA) bus. Designing, assembling, and testing this circuit board finally achieves O'Gorman's call for a digital humanist picking up a soldering iron as a non-trivial tutor text (Figure 10).

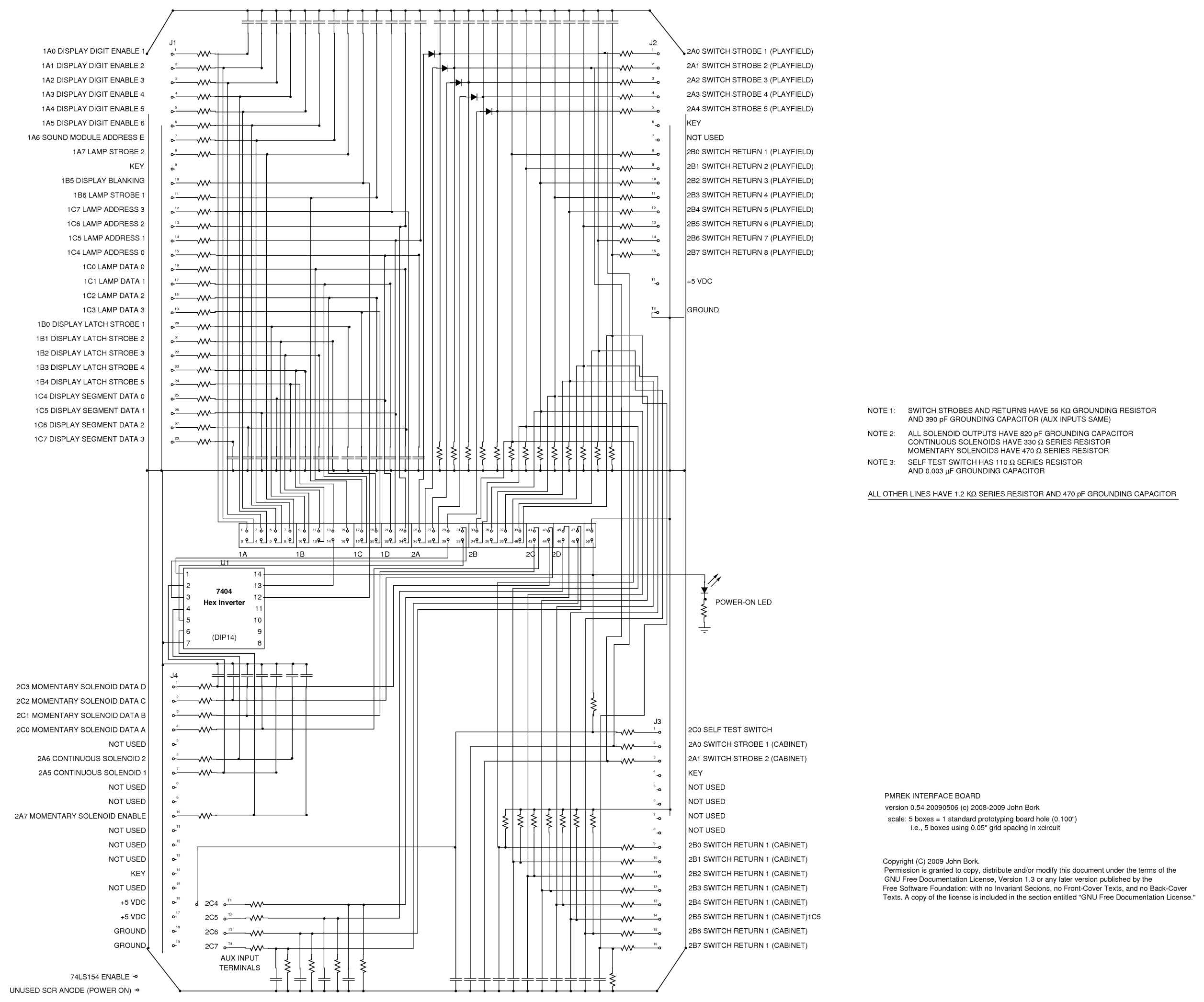

J1
1A0 DISPLAY DIGIT ENABLE 1
1A1 DISPLAY DIGIT ENABLE 2
1A2 DISPLAY DIGIT ENABLE 3
1A3 DISPLAY DIGIT ENABLE 4
1A4 DISPLAY DIGIT ENABLE 5
1A5 DISPLAY DIGIT ENABLE 6
1A6 SOUND MODULE ADDRESS E
1A7 LAMP STROBE 2
KEY
1B5 DISPLAY BLANKING
1B6 LAMP STROBE 1
1C7 LAMP ADDRESS 3
1C6 LAMP ADDRESS 2
1C5 LAMP ADDRESS 1
1C4 LAMP ADDRESS 0
1C0 LAMP DATA 0
1C1 LAMP DATA 1
1C2 LAMP DATA 2
1C3 LAMP DATA 3
1B0 DISPLAY LATCH STROBE 1
1B1 DISPLAY LATCH STROBE 2
1B2 DISPLAY LATCH STROBE 3
1B3 DISPLAY LATCH STROBE 4
1B4 DISPLAY LATCH STROBE 5
1C4 DISPLAY SEGMENT DATA 0
1C5 DISPLAY SEGMENT DATA 1
1C6 DISPLAY SEGMENT DATA 2
1C7 DISPLAY SEGMENT DATA 3
J2
2A0 SWITCH STROBE 1 (PLAYFIELD)
2A1 SWITCH STROBE 2 (PLAYFIELD)
2A2 SWITCH STROBE 3 (PLAYFIELD)
2A3 SWITCH STROBE 4 (PLAYFIELD)
2A4 SWITCH STROBE 5 (PLAYFIELD)
KEY
NOT USED
2B0 SWITCH RETURN 1 (PLAYFIELD)
2B1 SWITCH RETURN 2 (PLAYFIELD)
2B2 SWITCH RETURN 3 (PLAYFIELD)
2B3 SWITCH RETURN 4 (PLAYFIELD)
2B4 SWITCH RETURN 5 (PLAYFIELD)
2B5 SWITCH RETURN 6 (PLAYFIELD)
2B6 SWITCH RETURN 7 (PLAYFIELD)
2B7 SWITCH RETURN 8 (PLAYFIELD)
+5 VDC
GROUND
NOTE 1: SWITCH STROBES AND RETURNS HAVE 56 KΩ GROUNDING RESISTOR AND 390 pF GROUNDING CAPACITOR (AUX INPUTS SAME)
NOTE 2: ALL SOLENOID OUTPUTS HAVE 820 pF GROUNDING CAPACITOR CONTINUOUS SOLENOIDS HAVE 330 Ω SERIES RESISTOR MOMENTARY SOLENOIDS HAVE 470 Ω SERIES RESISTOR
NOTE 3: SELF TEST SWITCH HAS 110 Ω SERIES RESISTOR AND 0.003 μF GROUNDING CAPACITOR
ALL OTHER LINES HAVE 1.2 KΩ SERIES RESISTOR AND 470 pF GROUNDING CAPACITOR
1A
1B
1C
1D
2A
2B
2C
2D
U1
7404
Hex Inverter
(DIP14)
POWER-ON LED
J4
2C3 MOMENTARY SOLENOID DATA D
2C2 MOMENTARY SOLENOID DATA C
2C1 MOMENTARY SOLENOID DATA B
2C0 MOMENTARY SOLENOID DATA A
NOT USED
2A6 CONTINUOUS SOLENOID 2
2A5 CONTINUOUS SOLENOID 1
NOT USED
NOT USED
2A7 MOMENTARY SOLENOID ENABLE
NOT USED
NOT USED
NOT USED
KEY
NOT USED
+5 VDC
+5 VDC
GROUND
GROUND
2C4
2C5
2C6
2C7
AUX INPUT TERMINALS
74LS154 ENABLE
UNUSED SCR ANODE (POWER ON)
J3
2C0 SELF TEST SWITCH
2A0 SWITCH STROBE 1 (CABINET)
2A1 SWITCH STROBE 2 (CABINET)
KEY
NOT USED
NOT USED
NOT USED
NOT USED
2B0 SWITCH RETURN 1 (CABINET)
2B1 SWITCH RETURN 1 (CABINET)
2B2 SWITCH RETURN 1 (CABINET)
2B3 SWITCH RETURN 1 (CABINET)
2B4 SWITCH RETURN 1 (CABINET)
2B5 SWITCH RETURN 1 (CABINET)1C5
2B6 SWITCH RETURN 1 (CABINET)
2B7 SWITCH RETURN 1 (CABINET)
PMREK INTERFACE BOARD
version 0.54 20090506 (c) 2008-2009 John Bork
scale: 5 boxes = 1 standard prototyping board hole (0.100") i.e., 5 boxes using 0.05" grid spacing in xcircuit
Copyright (C) 2009 John Bork.
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation: with no Invariant Secions, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License."

Figure 10: Pmrek Interface Circuit Board to Existing Wiring

Here, more than anywhere else, the schematism that Kittler talks about comes to the fore. Layers of diachrony in synchrony are expressed by the physical electronic connections between the computer's Intel x86 architecture's Complimentary Metal-Oxide Semiconductor (CMOS) circuitry and that of the existing Bally pinball boards, the processes built into the Linux kernel workqueue, and the supervisory game control routine running as an ordinary application in user space called `testbed`.

Pinball Platform Studies

Montfort et. al. present *10 PRINT* as a foray into platform studies by examining the one line program token by token, taking each down to the platform level of the Commodore 64, and then extending the analysis to other computer architectures and variations of the maze drawing program. In their prior work *Racing the Beam*, Montfort and Bogost examined the Atari VCS platform using an approach “mainly informed by the history of material texts, programming, and computing systems” because “only the serious investigation of computing systems as specific machines can reveal the relationships between these systems and creativity, design, expression, and culture” (3-4). From my experience developing `pmrek`, I realized that I could perform a parallel analysis of the Bally pinball platform, inaugurating the subfield Pinball Platform Studies.

Concurrent to the first decade of videogames, hundreds of thousands of electronic pinball machines were created using a small number of discrete architectures among a handful of major manufacturers. The Bally AS 2518-35 exemplifies one such platform. What is it like? What affordances and constraints can be intuited from its design, and how do they play out in various games? How can the study of pinball platforms contribute to an overall awareness of computer technology, as well as their cultural and social milieu that is part of a recent past now forming the mythical foundation of the Internet age? What can we do with pinball as humanities scholars who also solder and write software? These are questions that spring directly from the declaration of platform studies given by Montfort and Bogost, and flesh out the theme of remediating electracy by engaging in a grey software project like `pmrek` to explore the layers of diachrony in synchrony characteristic of code/space.

The Bally AS 2518-35 platform was selected because more pinball machines were manufactured using this set of control boards than any other early electronic offering from the major manufacturers Bally, Williams, Gottleib, and Stern. The primary resources for learning about the Bally architecture are U.S. Patent 4,198,051, the *Bally Electronic Pinball Games Theory of Operation*, and the documentation created for each specific model, which typically include schematics of the various revisions of the circuit boards in the AS 2518-35 family it used. The patent describes the pinball machine platform in one very long sentence. “A pinball machine having a playing field with a plurality of ball responsive switches and indicator lights, a digital computer comprising means for supplying switch address signals corresponding to selected ball responsive switches, means for receiving input signals in response to said ball responsive switches and means for supplying output signals corresponding to selected indicator lights in response to said input signals, and a plurality of controlled switches respectively associated with said indicator lights for conducting power to illuminate the lights, said means for supplying output signals having an output port for supplying indicator light data signals and an output port for supplying indicator light address signals, said machine further having a plurality of decoders each having multiple input lines operably connected to the address signal output port, multiple output lines each being operably connected to a controlled switch, and an enable line operably connected to the data signal output port, each of said decoders being responsive to a data signal from the data signal output port to decode the address signals from the address signal output port and provide a signal to a controlled switch in accordance with the address from the address signal output port whereby a selected indicator light may be illuminated, and said machine further having control means including interface means having an output port and an input port and being operatively connected to the ball responsive switches for supplying switch addressing test signals to said selected switches in response to the computer means switch address signals and for supplying switch input signals from said selected switches to said input port" (Bracha et. al. np).

Like the VCS, pinball platforms used small 2K ROMs to store the low-level hardware routines and game programs. Where the specialized TIA integrated circuit (IC) that the VCS used to pace the beam of the television electron gun that drew a game's sprite images in real time, the key component of the Bally games are the ICs handling physical input and output between the microprocessor unit (MPU) and the other final control elements for the solenoids, switches, feature lamps, and digital displays. According to Montfort and Bogost, "the chip on the VCS board that handles most of the input from controllers is a standard one, a MOS Technology 6532. . . . Because of these three functions, the chip is called the RIOT (RAM/Input/Output/Timer); in the *Stella Programmer's Guide*, it is referred to as the Peripheral Interface Adapter (PIA)" (23). Bally pinball machines used a pair of the Motorola version 6820 PIA, and for `pmrek` two of the Intel equivalent 8255 Programmable Peripheral Interface (PPI) ICs. Thus for Bally pinball designs, the number of switch inputs, solenoid, and lamp, and display outputs are constrained by the 48 total digital input and output lines afforded by these two PIAs.

The *Theory of Operation* makes the basic comparison of machine control system components to human body perceptual systems, and program execution to cognition, informing the program as the senses do the brain to affect behavior. "The momentary switches in the matrix are the 'eyes' and 'ears' of the MPU chip. It is only by means of sensing closures, and later, reacting to valid closures (during normal operation) that the MPU, thru the program, knows what it is to do next!" (6). This document makes it clear that the design leverages the interesting capability of the Motorola 6800 microprocessor of being interrupted while it is servicing an interrupt. “The periodic interrupts are generated by the Zero Crossing Detector and the Display Interrupt Generator on the MPU module. The former occurs at a rate of 120 times per second, or once each power line zero crossing. The second occurs at a rate of approximately 320 times per second as determined by R21, R22 and C16” (7). Thus the moniker *constantly interrupted* cleverly characterizes the basic AS 2518-35 platform operation as pacing the beam does for the Atari VCS. For the deadlines and shorttimes of the surrounding circuitry to be satisfied using this model of cascading interruptions, “in these cases, the programmer must carefully cycle count processor instructions so they execute at the right time" (28).[124]

Through such primary resources and reverse engineering techniques detailed in my masters thesis, Pinball Platform Studies considers pinball operation, both under the original,

124 Rather than replicating this delicate, time-sensitive interrupt logic the depends on an external circuit to generate the interrupts, I devised a “shotgun method” for `pmrek` that used the relatively high frequency of the Linux kernel workqueue execution for triggering the silicon-controlled rectifiers (SCRs) driving the feature lamps, strobing the switch matrix, and counting the cycles to achieve the momentary solenoid firing duration of 26 ms.

special-purpose AS 2518-35 computing platform, and how it can be implemented using a newer, operating system controlled, nondeterministic, general purpose computer. The similarity to the VCS in the overall operation is initially striking as we begin to familiarize ourselves with how these machines work. Where “thanks to the TIA's provision for collision detection in hardware, it is easy to implement things such as shooting or being shot by missiles, running into a wall, or consuming something,” (48) in pinball various playfield mechanisms afford game types, with lots of variation in how they are utilized. Yet these physical features must be amenable to the basic control operations of the two continuous, sixteen momentary solenoids, forty matrixed switches, sixty-four matrixed lamps, and the five six-digit, seven-segment numeric displays the AS 2518-35 MPU supports. For instance, in the earliest electronic pinball machines, sounds were produced by a set of xylophone-like chimes each struck by a momentary solenoid. Later games redirected the solenoid output to an auxilliary sound board that produced digital music and even synthesized speech based on the values sent to it that were formerly solenoid numbers demultiplexed to individual transistor driver circuits. “Just as a practice like letterpress printing is a contemporary, ongoing activity in addition to being the dominant method of printing from times past, the Atari VCS is admirable for its historical role in video gaming while it remains playable and programmable today" (143). Pinball platforms share with the VCS opportunities for discovering additional platform capabilities, exploring creative computing, and learning programming.

Machine Embodiment

In the second chapter I discussed cyborg embodiment and how Hayles' studies of the Macy Conferences on cybernetics reveal an interplay between the models used to explain neuronal activity and the artifacts of mechanical and early electronic computation. Her contention is that the two intermixed early on, so that our conceptions of how our minds work are based on how those computers work, and vice-versa, explaining how we became posthuman a long time ago. A key point of second and third wave cybernetic theory, and contemporary theorists of extended cognition, is that the human mind is significantly embodied, not discretely separated like Leibniz's monad or Descartes' thinking thing. Furthermore, it is wholly intermingled with the built environment, the code/space of everyware. Thus there is every reason to accept von Neumann's recasting of the Socratic imperative 'know thyself' by looking to our second selves in our machines. "Of all automata of high complexity, computing machines are the ones which we have the best chance of understanding. In the case of computing machines the complications can be very high, and yet they pertain to an object which is primarily mathematical and which we understand better than we understand most natural objects" ("Theory and Organization of Complicated Automata" 435).

The problem is that the way we go about conceiving artificial automata is by basing computation on effective procedures, doing things that humans can be trained to do in 'moronic' fashion (Copeland's term, not mine). Flowcharts, process diagrams, source code are read by tracing the imagined movement of electrons to empower circuits that enact such procedures. Recall the Quicksort algorithm and more obvious examples of working code I have presented. Yet as Clark points out, "Ballard et al. (1997) suggest using the term the embodiment level to indicate the level at which functionally critical operations occur at timescales of around one-third second" (*Supersizing the Mind* 14). What happens at timescales beyond this threshold we imagine by speeding up the cycles, multiplying the number of discrete units, and multiplexing their involvement with parallel processes to form the distributed agency of everyware. While an imagined zooming to microscopic levels suffices for explanatory purposes—the programs and circuit work as designed, after all—it does not convey what it *feels like* to be such a machine. As Bogost argues, "to put things at the center of a new metaphysics also requires us to admit that they do not exist just for us" (5). Alien phenomenology implies deprivileging human perception as that which defines objects, accepting that all objects recede interminably into themselves.

Intuiting machine embodiment entails learning to read more than source code. It entails understanding the language of circuit schematics and the unit operations of electrical devices like resistors, capacitors, power supplies, and electronic devices like diodes, transistors, logic gates, and their compound assemblies in integrated circuits like the Motorola 6820 and Intel 8255, and the microprocessors that embody the stored program Burks, Goldstine, and von Neumann introduced in their report. These devices constituting the bottom, physical level of the network layer diagrams also embody the platform layer of Montfort and Bogost's five levels of digital media. Getting to know these circuits better helps retrieve these artificial automata from the simplistic reduction to familiar mechanisms that exist at the perceptual level like door latches and floodgates. Just as Galloway insisted on discerning traces of Marxian vitalism in networking protocols, I wish to further develop the ideas Guattari presents in "Machinic Heterogenesis" that I touched in the section on working code places. There he states that "although machines are usually treated as a subheading of technics, I have long thought that it was the problematic of technics that remained dependent on the questions posed by machines. Machinism is an object of fascination, sometimes of delirium. There exists a whole historical beastiary of things relating to machines" (13). His nice twist of assimilating the machine into living or living into machine seems like an act of philosophical finesse, and he and Deleuze provide many fascinating examples from these besatiaries in their writings.

Guattari seeks an exit from conceiving machines in strictly functionalist terms, as mechanisms that comply with their enumerated design principles. "The diverse modalities of machine autopoiesis essentially escape from signifying mediation and refuse to admit to any general syntax describing the procedures of deterritorialization" (16). While the exit he seeks for them lies in their inherent potential of "this autopoeitic nexus of the machine" to wrest itself from "structural retroactions, their input and output, [which] are called upon to function according to a principle of eternal return" (15), my argument hinges on the submersion of machinic activity into the nonsensical realms of alien temporalities and the thrust of critical programming to better understand what is really going on in these beastiaries. Thus Guattari hints at "another form of alterity has been taken up only very indirectly, one we could call the alterity of scale, or fractal alterity, which sets up a play of systematic correspondence among machines belonging to different levels" (21) that parallels the layers model diagram for procedural rhetorics of diachrony in synchrony. So here is a revision that includes the LAMP components and working code places from my three foss projects on the vertical continuum, as well as depicting the piling up of components of the theoretical framework in their respective domains with respect to culture and content (Figure 11).

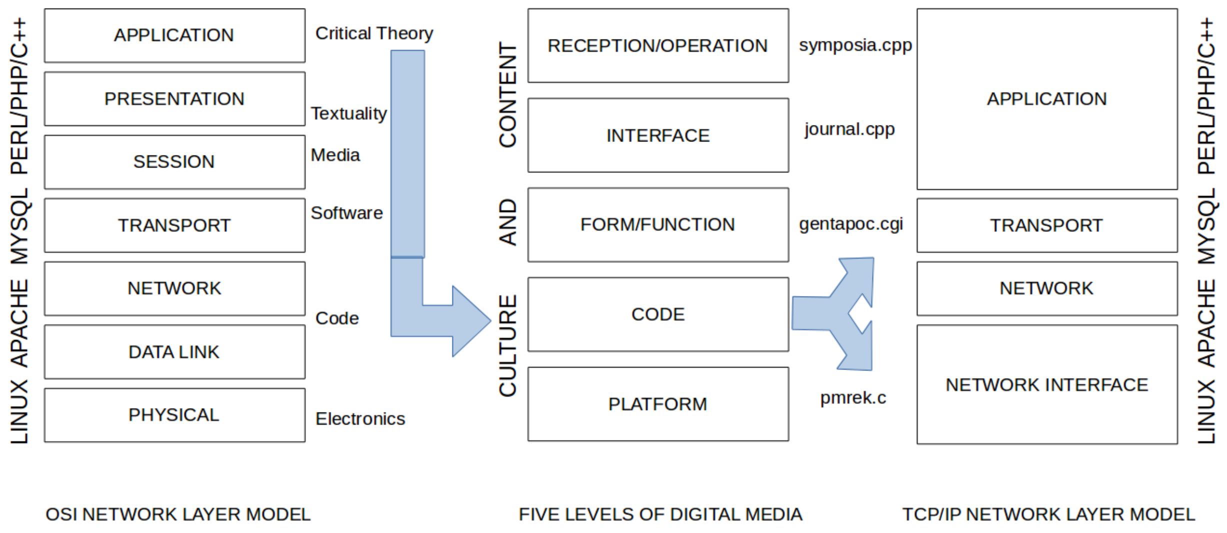


Figure 11: Revised Levels Models with Working Code Places

Turkle recalls a golden age of craft programming at the birth of microcomputers, “an age when a programmer was a skilled artisan who was given a problem and asked to conceive of and craft a solution,” whereas now “programs are written on a kind of assembly line. The professional programmer works as part of a large team and is in touch with only a small part of the problem being worked on” (*Second Self* 170). Working on isolated portions of a larger project supports the isolation of these heterogeneous, interrelating levels, confirming Kraft's suspicion that structured programming has been leveraged as a managerial tool for deskilling the “collective mythology of the shop” to which Turkle alludes. Moreover, adopting Guattari's terms it permits the discursive analysis of software components to operate in terms of the Lacanian signifier alone as originating from linguistic structuralism. Therefore, synonymous with linear discursivity, its ontological guarantee subsists only in movement from symbol to symbol, missing basic facets of the operation of heterogeneous machines. “Why this lack of reverence toward the Lacanian conception of the signifier? It is precisely because this theorization, coming out of linguistic structuralism, does not get us out of structure, and prohibits us from entering the real world of the machine. . . . Asignifying semiotic figures do not secrete only significations. They issue starting and stopping orders and, above all, they provoke the setting into being of ontological universes" (23-24).

Glancing back at *A Thousand Plateaus*, the authors describe the imagination required to become something else, like a woman or a dog, not as imitation but to "make your organism enter into composition with something else in such a way that the particles emitted from the aggregate thus composed will be canine as a function of the relation of movement and rest, or of molecular proximity, into which they enter" (274). They continue: "if becoming-woman is the first quantum, or molecular segment, with the becomings-animal that link up with it coming next, what are they all rushing toward? Without a doubt, toward becomings-imperceptible. The imperceptible is the immanent end of becoming, its cosmic formula" (279). Does becoming-imperceptible amount to intuiting machine embodiment, things happening in humanly incomprehensible temporal orders of magnitude and humanly impossible to enumerate quantities, following their logic that the first radicalization is becoming the other sex, as that which most popularly distinguishes humans, then becoming animals, then single atoms, down to the smallest particle, is this where the crossovers between human and machine cognition occur?

CHAPTER 5: CONCLUSION

Critical programming studies technical devices through working code as a digital humanities practice. Working code as a verb is *practice*, in the sense of the human activity of working with code, including creating it, testing it, and ultimately using it in the form of running software; as noun it is the *source* of running software, as that whose compilation or interpretation becomes a part of those machine instructions constituting computation, and programming language 'texts' of human consideration, as that which is the object of human attention when referring to 'code'. Few critical theorists addressing the human situation with respect to technological media more compellingly evoke the serious need to study it, while at the same time harboring a fundamentally pessimistic outlook on the endeavor, than Friedrich Kittler in the preface to *Gramophone, Film Typewriter*. Even in translation, his argument that we have long passed the point of being able to take stock of our condition is doubly convincing, for its rhetoric appeals both to those who are ignorant of the technological details of the schematism of perceptibility of technological media, and, it seems, more strongly prohibitive for those who do. That is, reaching the Socratic wisdom of knowing that we cannot know everything or even enough about how it all works to critically engage it as a whole; at best attaining only working knowledge of small domains as software engineers do particular source code bases and specifications, we rationally decide to foreclose further, focused study of TCP/IP or C++, for example.

Yet it has always been the task of philosophers and humanists to evaluate the human condition. We cannot shun study of our machine others that are also our second selves and maintain understanding of how they work. Decades of thinking and thoughtfully writing software has brought me to conclude that there is, and has always been, a crisis at the heart of the philosophy of computing, namely the dilemma between the incompatible epistemological imperative to comprehend the subject matter, and the reflexive avoidance to *not* look too deeply at it, to *not* study what is judged to be inscrutable and ephemeral anyway. This response echoes Seneca, explaining why the sage does not trifle in engineering knowledge, not to pick up that which must be laid aside (*ponenda non sumeret*), retransmitted through Ong that we not consider programming languages in the same breath as any mother tongue. The consequence of this avoidance, reminiscent of the Heideggerian retreat of being in the essence of modern technology, is that our intellectual trajectory has deaccelerated from symbiotic improvement as cyborgs intermediated with machine technologies to humans getting dumber, like the Nietzschean last man and bourgeois capitalist commodity consumers of Horkheimer and Adorno, while the machines continue to get smarter. To begin to extricate ourselves from this downward spiral is a the task for the philosophy of computing; humanities thinking moves toward it when it braves to not turn away from technological knowledge. We can fix this. By developing a more finely nuanced conception of the human situation with respect to technological media by considering their constitutive platforms through the method of critical programming studies, rekindling interest in learning programming among adults and promoting its learning as any other essential human skill like orality and literacy for future generations, humanity and machinery may alter their collective, long term intellectual course.

Cyborg Revisited

It is time to revisit the post-postmodern network dividual cyborg depicting the as-is situation under technological media and global capitalism to see how excursions into critical programming in working code places may alter its trajectory. Turkle frames the problem well by explaining that the modernist, Enlightenment depth of early command line oriented personal computing was supplanted by the opaque, interface level technology of the Macintosh, only to be remixed into hybridity with Microsoft Windows. Since the turn of the century, entirely free, open source operating systems are available for exploration and everyday use that expose a rabbit hole back down into the depths of computationalism spanning the operation of running applications as well as the network protocols sustaining cyberspace. This epistemological transparency is threatened by a resurgence of proprietary software and protocols that proliferate particularly in mobile devices, as well as the code/space around us that is shot through with its own secret commands. As Kitchin and Dodge put it, “from Dumb to Smart . . . In this context, smart means programmed awareness of use, rather than intelligence. . . . The car is prime example of an assemblage that is rapidly shifting from inert machine to aware object” (99).

The recent revelation that millions Volkswagen automobile control computers were deliberately programmed to detect and cheat emissions tests is a stark reminder of how much faith we put in these mysterious black boxes to do what their makers promise they do. Haraway's analogy of the American Southwest India coyote/trickster “suggests our situation when we give up mastery but keep searching for fidelity, knowing all the while we will be hoodwinked,” while “our hopes for accountability in the techno-biopolitics in postmodern frames turn on revisioning the world as coding trickster with whom we must learn to converse” (199; 209). This conversation with the coding trickster may thus take on a number of basic forms.

The default form of human machine comportment seems to be that of the surfer who rides atop the digital streams, exercising skill in navigating the waters, and even making small contributions to their motions, without understanding all the science behind wave mechanics or the chemistry of the fluids. Manovich adopts Geert Lovink's Data Dandy, with a nod to Benjamin's flaneur walking the arcades (*Language of New Media* 270). Floridi calls it the 'produmer', acknowledging the combination of consumption and production, but makes the point that “unless responsible individuals, as well as academic, cultural institutions or recognized organizations, provide some forms of guides, filters, selective agents and authoritative services (e.g. rating services, quality-control services), we may no longer be able to distinguish between the intellectual space of information and knowledge and a highly polluted environment of junk mail and meaningless data” (85).

A more nuanced relationship to the computational substrate of everyware is offered by Jenkin's monitorial citizen, whose productive engagement “is not simply being able to read and write, but being able to participate in the deliberations over what issues matter, what knowledge counts, and what ways of knowing command authority and respect” (269). This figure knows how to use social media and other technological resources wisely, in this new conception of wisdom that Berry presents via Serres' parasite: being a good stream, “who this subject 'eats next to'” (170). Kitchin and Dodge and others call it Web 2.0, whose distinct phase of online production that has rapidly become embedded in everyday life affording enhanced degrees of agency to users and creators, for example blogging and other new forms of participatory dialog, and mashups adding value to harvested capta (125). At the same time, people voluntarily provide huge amounts of capta through ordinary interactions, and in turn receive an endless stream of 'free', personalized content, positioning them for a *WALL-E* future on the *Axiom* under the watchful master control AUTO—or worse.[125]

Crowd-sourced and amateur gazes are now complemented by Wikileaks and other dissemination of official documents via insiders and contractors. Yet there is still a prejudice that the human/machine border shimmering at our sensory threshold is as impenetrable and alien to us as our intelligence, cognition, and emotions are to devices. However, Hayles' research suggests that embodiment precedes cogitation, turning Descartes upside down (*How We Became Posthuman* 203). For her "becoming a posthuman means much more than having prosthetic devices grafted onto one's body. It means envisioning humans as information-processing machines with fundamental similarities to other kinds of information-processing machines, especially intelligent computers. . . . Increasingly the question is not whether we will become posthuman, for posthumanity is already here. Rather, the question is what kind of posthumans we will be" (246). As I have suggested through the experiments with critical programming, we may learn to intuit some of these forms of machine embodiment in high speed processes and unit operations by spending a significant amount of time in working code places.

Latour goes further when he proposes, as his own answer to "Why Has Critique Run Out of Steam?" a super-critical constituent of collective intelligence; a transpersonal, post-

125 AUTO appears in the movie as a ship's wheel that wields control over the *Axiom* spaceship in a gallery of portraits of successive generations of its human captains. Dare we imagine a similar image in which a number of famous and infamous hardware engineers and programmers are posed beside the same Hollerith punch card machine rescued from a concentration camp, like the good Father Busa and the Nazi villains from the suppressed IBM *History of Computing in Europe*?

postmodern subjectivity that is extended into machines and objects. Multifarious inquiry is required to detect the participants gathered in a thing. “The solution lies, it seems to me, in this promising word *gathering* that Heidegger had introduced to account for the 'thingness of the thing'. . . . Whatever the words, what is presented here is an entirely different attitude than the critical one, not a flight into the conditions of possibility of a given matter of fact, not the addition of something more human that the inhumane matters of fact would have missed, but, rather, a multifarious inquiry launched with the tools of anthropology, philosophy, metaphysics, history, sociology to detect how many participants are gathered in a thing to make it exist and to maintain its existence. Objects are simply a gathering that has failed—a fact that has not been assembled according to due process” (245-246). This approach aligns with Harman, Thrift, Bogost, and others who propose object-oriented, variable ontologies that fit into my diachrony in synchrony diagram of cyberspace.

It is fitting that in his essay Latour invokes Turing's musings on computing machinery and intelligence to imagine this super-critical intelligence, piled up like neutrons going critical, yielding a thinking machine: “Here Turing too cannot avoid mentioning God's creative power when talking of this most mastered machine, the computer that he has invented. . . . In the most dramatic way, Turing's paper demonstrates, once again, that all objects are born things, all matters of fact require, in order to exist, a bewildering variety of matters of concern. The surprising result is that we don't master what we, ourselves, have fabricated, the object of this definition of critique” (247). Yet seldom do users reprogram the underlying software to affect production and consumption, although it is always held out as a democratizing option. Therefore, I wish to conclude by relating this notion of piling up to my syncretism of layers in the theoretical framework, with the hope that it a critical mass is reached in enough digital humanists who also pick up soldering irons and write code so that new philosophical places explode on the scene.

Toward super-critical theory: is the black box, input output analysis partially to blame for prevalence of devalued objects over field effects, matters of concern, things? Do we too quickly trust that the double mediation performed by digital devices is redeemed by direct manipulation? Latour's parting invocation to please touch and deploy ties O'Gorman's scholarly remainders harboring diachrony-within-synchrony and Bogost's philosophical carpentry into critical programming studies, so I shall make it mine as well. “We all know subcritical minds, that's for sure! What would critique do if it could be associated with *more*, not with *less*, with *multiplication*, not with *subtraction*. Critical theory died away long ago; can we become critical again, in the sense here offered by Turing? That is, generating more ideas than we have received, inheriting from a prestigious critical tradition but not letting it die away. . . . This would require that all entities, including computers, cease to be objects defined simply by their inputs and outputs and become again things, mediating, assembling, gathering many more folds than the 'united four'. If this were possible then we could let the critics come ever closer to the matters of concern we cherish, and then at last we could tell them: 'Yes, please, touch them, explain them, deploy them'. Then we would have gone for good beyond iconoclasm” (248).

LIST OF REFERENCES

Aarseth, Espen J. “Nonlinearity and Literary Theory.” Eds. Noah Wardrip-Fruin and Nick Montfort. *The New Media Reader*. Cambridge, Mass: MIT Press, 2003. 762-780. Print.

Abbate, Janet. *Inventing the Internet*. Cambridge, MA: The MIT Press, 1999. Print.

Adorno, Theodor W. “On the Fetish-Character in Music and the Regression of Listening.” *Essays on Music*. Ed. Richard Leppert. CA: University of California Press, 2002. Print.

Aloisio, Mario. "The Calculation Of Easter Day, And The Origin And Use Of The Word Computer." *IEEE Annals Of The History Of Computing* 26.3 (2004): 42-49. Web. 6 Dec. 2012.

Applen, J D, and Rudy McDaniel. *The Rhetorical Nature of Xml: Constructing Knowledge in Networked Environments*. New York: Routledge, 2009. Print.

Barthes, Roland. “Listening.” *The Responsibility of Forms*. Ed. Richard Howard. CA: University of California Press, 1985. Print.

---. *Mythologies*. Trans. Annette Lavers. New York: The Noonday Press, 1972. Print.

---. “Myth Today.” *A Barthes Reader*. Ed. Susan Sontag. New York: Barnes and Noble, Inc., 2009. Print.

---. “The Structuralist Activity.” Eds. Richard DeGeorge and Fernande DeGeorge. *The Structuralists from Marx to Levi-Strauss*. Garden City, New York: Doubleday, 1972. 149-154. Print.

Barthes, Roland and Stephen Heath. *Image, Music, Text*. New York: Hill and Wang, 1977. Print.

Baudrillard, Jean. *Simulacra And* Simulation. Trans. Sheila Faria Glaser. Ann Arbor, MI: University of Michigan Press, 1994. Print.

---. *The Transparency of Evil: Essays on Extreme Phenomena*. Trans. James Benedict. New York: Verso, 1993. Print.

Bauerlein, Mark. *The Dumbest Generation: How the Digital Age Stupefies Young Americans and Jeopardizes Our Future*. New York: Jeremy P. Tarcher/Penguin, 2009. Print.

Benjamin, Walter. "The Work of Art in the Age of Mechanical Reproduction." *Film Theory and Criticism: Introductory Readings*. New York: Oxford University Press, 1999. 731–751. Print.

Berry, David M. *The Philosophy of Software: Code and Mediation in the Digital Age*. Basingstoke, Hampshire: Palgrave Macmillan, 2011. Print.

Bijker, Wiebe E., Thomas P. Hughes, and Trevor Pinch, eds. *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Cambridge, MA: The MIT Press.,1987. Print.

Black, Edwin. *IBM and the Holocaust: The Strategic Alliance Between Nazi Germany and America's Most Powerful Corporation*. New York: Crown Publishers, 2001. Print.

Bogost, Ian. *Alien Phenomenology: or What It's Like to Be a Thing*. Minneapolis: University of Minnesota Press, 2012. Print.

---. *Persuasive Games: The Expressive Power of Videogames*. Cambridge, MA: MIT Press, 2007. Print.

---. *Unit Operations : an Approach to Videogame Criticism.* Cambridge : MIT Press, 2006. Print.

Boltanksi, Luc and Eve Chiapello. *The New Spirit of Capitalism*. Trans. Gregory Elliott. New York: Verso, 2005. Print.

Bolter, J. David. *Writing Space : Computers, Hypertext, and the Remediation of Print*. 2nd ed. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2001. Print.

Bork, John. *Reverse Engineering a Microcomputer-based Control Unit*. MIT Thesis. Bowing Green State University, Bowling Green, Ohio, 2005. OhioLINK Electronic Thesis and Dissertation Center. Web. 27 Sept. 2015.

Bracha, et al. “Computerized pin ball machine.” Patent 4,198,051. 15 April 1980.

Brin, David. “Why Johnny can't code”. *Salon* 14 Sept. 2006. Web. 1 Apr. 2013.

Brooks, Jr., Frederick P. *The Mythical Man-Month: Essays on Software Engineering Anniversary Edition*. Boston: Addison Wesley Longman, Inc., 1995. Print.

Brown, John Seely and Paul Duguid. *The Social Life of Information.* Boston: Harvard Business School Press, 2000. Print.

Buck-Morss, Susan. “Dream World of Mass Culture.” *Modernity and the Hegemony of Vision*. Ed. David Michael Levin. Berkeley: University of California, 1993. 309-335. Print.

Burnard, Lou, Katherine O’Brien O’Keeffe, and John Unsworth. *Electronic Textual Editing.* New York: Modern Language Association of America, 2006. Print.

Burks, Arthur W., Herman H. Goldstine and John von Neumann. “Preliminary Discussion of the Logical Design of an Electronic Computing Instrument.” U.S. Army Ordnance Department, 1946. Print.

Burnett, Ron. *How Images Think*. Cambridge, Mass: MIT Press, 2004. Print.

Busa, Roberto A. “Foreword: Perspectives on the Digital Humanities.” *A Companion to Digital Humanities*. Eds. Susan Schreibman, Ray Siemens, and John Unsworth. Malden, MA: Blackwell, 2004. xvi-xxi. Print.

Bush, Vannevar. “As We May Think.” *The New Media Reader*. Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 37-47. Print.

Bynum, Terrell Ward Bynum and Simon Rogerson. “Editors' Introduction: Ethics in the Information Age.” eds. Terrell Ward Bynum and Simon Rogerson. *Computer Ethics and Professional Responsibility*. Malden, MA: Blackwell, 2004. Print.

Callon, Michel. “Society in the Making: The Study of Technology as a Tool for Sociological Analysis.” *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Eds. Wiebe E. Bijker, Thomas P. Hughes, and Trevor Pinch. Cambridge, MA: The MIT Press.,1987. 83-103. Print.

Campbell-Kelly, Martin. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. Cambridge, Mass: MIT Press, 2003. Print.

Campbell-Kelly, Martin and William Aspray. *Computer: A History of the Information Machine*. New York: Basic Books, 1996. Print.

Castells, Manuel. *The Rise of the Network Society*. Oxford: Blackwell Publishers, 2000. Print.

C. E. W. *Bally Electronic Pinball Games Theory of Operation*. Chicago: Bally Corporation, 1982. Print.

Chun, Wendy Hui Kyong. "On 'Sourcery,' Or Code As Fetish." *Configurations: A Journal Of Literature, Science, And Technology* 16.3 (2008): 299-324. Web. 12 Dec. 2012.

---. *Programmed Visions: Software and Memory*. Cambridge, MA: The MIT Press, 2011. Print.

Clark, Andy. “An embodied cognitive science?” *Trends in Cognitive Sciences* 3:9 (September 1999): 345-51. Print.

---. *Supersizing the Mind: Embodiment, Action, and Cognitive Extension*. New York: Oxford University Press, 2008. Print.

Clark, Andy and David Chalmers. “The Extended Mind.” *Analysis* 58.1 (January 1998). 7-19. Print.

Conley, Verena Andermatt. “Preface.” *Rethinking Technologies*. Eds. Verena Andermatt Conley. Minneapolis: University of Minnesota Press, 1993. ix-xiv. Print.

Connor, Steven. “The Modern Auditory I.” *Rewriting the Self:Histories from the Renaissance to the Present*. Ed. Roy Porter. New York: Routledge. 1997. 203-223. Print.

Copeland, B. Jack. "What Is Computation?" *Synthese* 3 (1996): 335-357. Web. 6 Dec. 2012.

Cummings, Robert E. "Coding With Power: Toward A Rhetoric Of Computer Coding And Composition." *Computers And Composition* 23.(n.d.): 430-443. Web. 4 Jan. 2013.

De Saussure, Ferdinand. *Course in General Linguistics*. Trans. Roy Harris. Chicago: Open Court, 1986. Print.

Deleuze, Gilles. "Postscript on the Societies of Control." *October* (1992): 3-7. Web. 4 Jan. 2013.

Deleuze, Gilles and Felix Guattari. *A Thousand Plateaus: Capitalism and Schizophrenia*. Minneapolis: University of Minnesota Press, 1988. Print.

---. *What Is Philosophy?* New York: Columbia University Press, 1994. Print.

Derrida, Jacques. *Archive Fever: A Freudian Impression*. Chicago: University of Chicago Press, 1996. Print.

---. *Dissemination*. Chicago: University Press, 1981. Print.

---. *Of Grammatalogy*. Trans. Gayatri Chakravorty Spivak. Baltimore: The John Hopkins University Press, 1976. Print.

Diogenes Laertius. *Lives and Opinions of Eminent Philosophers in Ten Books.* Trans. R. D. Hicks. New York: G. P. Putnam's Sons, 1925. Print.

Dumit, Joseph. *Picturing Personhood: Brain Scans and Biomedical Identity*. Princeton, N.J: Princeton University Press, 2004. Print.

Dyson, Francis. “The Ear That Would Hear Sounds In Themselves: John Cage 1945-1965.” *Wireless Imagination: Sound, Radio and the Avane-Garde*. Eds. Douglas Kahn and Gregory Whitehead. Cambridge, MA: MIT Press, 1992. 373-401. Print.

Edwards, Paul N. *The Closed World: Computers and the Politics of Discourse in Cold War America*. Cambridge, MA: The MIT Press, 1996. Print.

Engelbart, Douglas. “Augmenting Human Intellect: A Conceptual Framework.“ *The New Media Reader*. Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 95-108. Print.

Ensmenger, Nathan. *The Computer Boys Take Over: Computers, Programmers, and the politics of technical expertise*. Cambridge: MIT Press, 2010. Print.

Feenberg, Andrew. “Democratic Rationalization: Technology, Power and Democracy.” *Technology and the Human Condition: A Philosophy of Technology Reader*. Eds. R. Scharff and V. Dusek. Malden, MA: Blackwell, 2002. 652-665. Print.

---. *Questioning Technology*. New York: Routledge, 1999. Print.

---. *Transforming Technology: A Critical Theory Revisited*. New York: Oxford University Press, 2002. Print.

Feller, Joseph, et al. *Perspectives on Free and Open Source Software*. Cambridge, MA: The MIT Press, 2005. Print.

Floridi, Luciano. *Philosophy and Computing: An Introduction*. London: Routledge, 1999. Print.

Foucault, Michel. *The Birth of Biopolitics: Lectures at the College de France 1978-1979*. Trans. Graham Burchell. New York: Palgrave Macmillan, 2008. Print.

---. *Discipline and Punish: The Birth of the Prison*. New York: Vintage Books, 1995. Print.

Frasca, Ganzalo, “Simulation versus Narrative.” *The Video Game Theory Reader*. Eds. Mark Wolf and Bernard Perron. New York: Routledge, 2003. 221-233. Print.

Freiberger, Paul and Michael Swaine. *Fire in the Valley: The Making of the Personal Computer*. 2nd ed. New York: McGraw-Hill. 2000. Print.

Fuller, Matthew. *Behind the Blip: Essays on the Culture of Software*. Brooklyn, NY, USA: Autonomedia, 2003. Print.

Fuller, Matthew, Ed. *Software Studies: A Lexicon*. Cambridge, MA: MIT Press, 2008. Print.

Galloway, Alexander R. *Protocol: How Control Exists After Decentralization.* Cambridge: MIT Press, 2004. Print.

Gane, Nicholas. “Computerized Capitalism: The Media Theory of Jean-Francois Lyotard.” *Information, Communication and Society* 6.3 (2003): 430-450. Print.

Gates, William H. *The Road Ahead*. New York: Viking, 1995. Print.

Gee, James Paul. *What Video Games Have to Teach Us About Learning and Literacy.* 2nd ed. New York: Palgrave Macmillan, 2007. Print.

Golumbia, David. *The Cultural Logic of Computation*. Cambridge: Harvard University Press, 2009. Print.

Goodman, Steve. *Sonic Warfare: Sound, Affect, and the Ecology of Fear*. Cambridge, Mass: MIT Press, 2010. Print.

Guattari, Felix. “Machinic Heterogenesis.” Trans. James Creech. *Rethinking Technologies*. Eds. Verena Andermatt Conley. Minneapolis: University of Minnesota Press, 1993. 13-26. Print.

Hafner, Katie and Matthew Lyon. *Where Wizards Stay Up Late: The Origins Of The Internet*. New York: Simon & Schuster, 1996. Print.

Hansen, Mark B. N. *Bodies in Code: Interfaces with Digital Media*. New York: Routledge, 2006. Print.

Haraway, Donna J. *Simians, Cyborgs, and Women: The Reinvention of Nature*. New York: Routledge, 1991. Print.

Hardt, Michael and Antonio Negri. *Empire*. Cambridge, MA: Harvard University Press, 2001. Print.

Harman, Graham. “On Vicarious Causation.” *Collapse* 2 (March 2007): 187-221. Web. 1 Jul. 2012.

Harper, Tauel. “Smash the Strata! A Programme for Techno-Political (r)evolution.” *Deleuze and New Technology*. Eds. David Savat and Mark Poster. Edinburgh: Edinburgh University Press, 2005. 125-140. Print.

Havelock, Eric. *The Muse Learns to Write: Reflections on Orality and Literacy from Antiquity to the Preset*. New Haven, CT: Yale University Press, 1986. Print.

Hayles, N. Katherine. *Electronic Literature*. Notre Dame: University of Notre Dame Press, 2008. Print.

---. *How We Became Posthuman: Virtual Bodes in Cybernetics, Literature, and Informatics*. Chicago: University of Chicago Press, 1999. Print.

---. *How We Think: Digital Media and Contemporary Technogenesis*. Chicago: The University of Chicago Press, 2012. Print.

---. *My Mother Was a Computer: Digital Subjects and Literary Texts*. Chicago: University of Chicago Press, 2005. Print.

---. "Print Is Flat, Code Is Deep: The Importance of Media-Specific Analysis." *Poetics Today* 25.1 (2004): 67-90. Web. 10 Apr. 2011.

---. *Writing Machines*. Cambridge, MA: The MIT Press, 2002. Print.

Heidegger, Martin. *An Introduction to Metaphysics*. Trans. Ralph Manheim. Garden City, NY: Anchor Books, 1961. Print.

---. *Nietzsche, Volume IV: Nihilism*. Trans. Joan Stambaugh, David Farrell Krell and Frank A. Capuzzi. San Francisco: Harper and Row, 1982. Print.

---. *The Question Concerning Technology, and Other Essays*. New York: Harper and Row, 1977. Print.

---. *What is Called Thinking?* Trans. J. Glenn Gray. New York: Harper and Row Publishers, 1968. Print.

Heilbroner, Robert L. “Do Machines Make History?” *Technology and Culture: An Anthology*. Eds. Melvin Kranzberg and William H. Davenport. New York: New American Library, 1972. 28-40. Print.

Heim, Michael. "The Computer as Component: Heidegger and McLuhan." *Philosophy And Literature* 16.2 (1992): 304-319. Print.

---. *Electric Language: A Philosophical Study of Word Processing*. New Haven: Yale University Press, 1987. Print.

Himanen, Pekka. *The Hacker Ethic and the Spirit of the Information Age*. New York: Random House, 2001. Print.

Hockey, Susan. "The History of Humanities Computing." *A Companion to Digital Humanities*. Eds. Susan Schreibman, Ray Siemens and John Unsworth. Malden, MA: Blackwell, 2004. 3-19. Print.

Horkheimer, Max and Theodor W. Adorno. *Dialectic of Enlightenment: Philosophical Fragments*. Stanford: Stanford University Press, 2002. Print.

Ihde, Don. *Philosophy of Technology: An Introduction*. New York: Paragon House, 1993. Print.

Iser, Wolfgang. *How To Do Theory*. Malden, MA: Blackwell Publishing, 2006. Print.

Jameson, Fredric. *Postmodernism, or, The Cultural Logic of Late Capitalism*. Durhman, NC: Duke University Press, 1991. Print.

Janz, Bruce B. *Philosophy in an African Place*. Lantham, MD: Lexington Books, 2009. Print.

---. "The Betweenness of Code." Unpublished presentation. 2012. Web.

---. "Reason and Rationality in Eze's *On Reason*." *South African Journal of Philosophy* 27.4 (2008): 296-309. Print.

Jenkins, Henry. *Convergence Culture: Where Old and New Media Collide*. New York: New York University Press, 2006. Print.

Johnson, Deborah. *Computer Ethics*. Engelwood Cliffs, NJ: Prentice-Hall, Inc., 1985. Print.

---. *Computer Ethics*. 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2001. Print.

---. *Computer Ethics: Analyzing Information Technology*. 4th ed. Upper Saddle River, NJ: Prentice Hall, 2009. Print.

Johnson, Richard. “What is Cultural Studies Anyway?” *Social Text* 16 (Winter 1986-1987): 38-80. Web. 1 Nov. 2011.

Johnston, John., ed. *Literature, Media, Information Systems: Friedrich A. Kittler Essays*. Amsterdam: Overseas Publishers Association, 1997. Print.

Kahn, Douglas. *Noise, Water, Meat: A History of Sound in the Arts*. Cambridge, Mass: MIT Press, 1999. Print.

Keller, Evelyn Fox and Christine R. Grontkowski. “The Mind's Eye.” *Discovering Reality: Feminist Perspectives on Epistemology, Metaphysics, Methodology and the Philosophy of Science*. Eds. Sandra Harding and Merrill Hintikka. Dordecht, Holland: Reidel Publishing, 1983. 207-221. Print.

Kellner, Douglas. "Critical Theory Today: Revisiting The Classics." *Theory, Culture & Society* 10.2 (1993): 43-60. *Alternative Press Index*. Web. 30 Sept. 2013.

Kemeny, John G. *Man and the Computer*. New York: Charles Scribner's Sons, 1972. Print.

Kemeny, John G. and Thomas E. Kurtz. *Back to Basic: The History, Corruption, and Future of the Language*. Reading, MA: Addison-Wesly Publishing Company, Inc., 1985. Print.

Kernighan, Brian W. and Dennis M. Ritchie. *The C Programming Language*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978. Print.

Kirschenbaum, Matthew G. “Extreme Inscription: Towards a Grammatology of the Hard Drive.” *Text Technology* 13.2 (2004): 91-125. Web. 12 Dec 2012.

Kitchin, Rob and Dodge, Martin. *Code/Space: Software and Everyday Life*. Cambridge, MA: MIT Press, 2011. Print.

Kittler, Friedrich A. *Discourse Networks 1800/1900*. Stanford: Stanford University Press, 1990. Print.

---. “Dracula's Legacy.” *Literature, Media, Information Systems: Friedrich A. Kittler Essays*. Ed. John Johnston. Amsterdam: Overseas Publishers Assocation, 1997. 50-84. Print.

---. *Gramophone, Film, Typewriter*. Stanford: Stanford University Press, 1999. Print.

---. “Protected Mode.” *Literature, Media, Information Systems: Friedrich A. Kittler Essays*. Ed. John Johnston. Amsterdam: Overseas Publishers Assocation, 1997. 156-168. Print.

---. “There Is No Software.” *Literature, Media, Information Systems: Friedrich A. Kittler Essays*. Ed. John Johnston. Amsterdam: Overseas Publishers Assocation, 1997. 147-155. Print.

---. “The World of the Symbolic—A World of the Machine .” *Literature, Media, Information Systems: Friedrich A. Kittler Essays*. Ed. John Johnston. Amsterdam: Overseas Publishers Assocation, 1997. 130-146. Print.

Knuth, Donald E. *Literate Programming*. Leland Stanford Junior University: Center for the Study of Language and Information, 1992. Print.

---. *Selected Papers on Computer Science*. Leland Stanford Junior University: Center for the Study of Language and Information, 1996. Print.

Knuth, Donald E. and Luis Trabb Pardo. “The Early Development of Programming Languages.” Stanford, CA: Stanford University Computer Science Department, 1976. Print.

Koerner, Brendan I. “Reading', Writin' and Ruby on Rails: Let's Teach Our Kids to Code.” *Wired* Oct. 2013. 29-32. Print.

Kraft, Philip. *Programmers and Managers: The Routinization of Computer Programming in the United States*. New York: Springer-Verlag, 1977. Print.

Krämer, Sybille. “The Cultural Techniques of Time Axis Manipulation: On Friedrich Kittler's Conception of Media.” Theory, Culture and Society 23.7-8 (December, 2006): 93-109. Web. 1 Aug. 2012.

Kurzweil, Ray. *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*. New York: Penguin Books, 1999. Print.

Lammers, Susan. *Programmers at Work: Interviews with 19 Programmers Who Shaped the Computer Industry*. Redmond, WA: Tempus, 1989. Print.

Landow, George P. *Hypertext 3.0: Critical Theory and New Media in an Era of Globalization*. Baltimore, MD: Johns Hopkins University Press, 2006. Print.

Lanier, Jaron. *Who Owns the Future?* New York: Simon and Schuster, 2013. Print.

Latour, Bruno. *Aramis, or The Love of Technology*. Trans. Catherine Porter. Cambridge, MA: Harvard University Press, 1996. Print.

---. *We Have Never Been Modern*. Cambridge, Mass: Harvard University Press, 1993. Print.

---. "Why Has Critique Run Out Of Steam? From Matters Of Fact To Matters Of Concern." *Critical Inquiry* 2 (2004): 225-248. Web. 2 Dec. 2012.

Leorke, Dale. “Rebranding the platform: The limitations of ‘platform studies.’” *Digital Culture and Education* 4.3 (2012): 257-268. Web. 1 June 2013.

Lessig, Lawrence. *Code version 2.0*. New York: Basic Books, 2006. Print.

---. *Free Culture: The Nature and Future of Creativity*. New York: Penguin Books, 2004. Print.

Levin, David M. “Introduction.” *Modernity and the Hegemony of Vision*. Ed. David M. Levin. Berkeley: University of California Press, 1993. 1-27. Print.

Lévy, Pierre. *Collective Intelligence: Mankind's Emerging World in Cyberspace*. Trans. Robert Bononno. Cambridge, MA: Perseus Books, 1997. Print.

Levy, Stephen. *Hackers: Heroes of the Computer Revolution*. New York: Bantam, 1984. Print.

---. *Insanely Great: The Life and Times of Macintosh, the Computer that Changed Everything*. New York: Penguin Books, 2000. Print.

Licklider, J. C. R. “Man-Computer Symbiosis.” *The New Media Reader*. Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 74-81. 2003. Print.

Lyotard, Jean-François. *The Postmodern Condition: A Report on Knowledge*. Minneapolis: University of Minnesota Press, 1984. Print.

---. *The Inhuman: Reflections on Time*. Trans. Geoffrey Bennington and Rachel Bowlby. Stanford, CA: Stanford University Press, 1991. Print.

Mackenzie, Adrian. *Cutting Code: Software and Sociality*. New York: Peter Lang Publishing, Inc. 2006. Print.

Malabou, Catherine. *What Should We Do with Our Brain?* Trans. Sebastian Rand. New York: Fordham University Press, 2008. Print.

Maner, Walter. "Unique Ethical Problems In Information Technology." *Science And Engineering Ethics* 2.2 (1996): 137-154. Web.17 Apr. 2013.

Manovich, Lev. *The Language of New Media.* Cambridge: MIT Press, 2001. Print.

---. *Software Takes Command*. New York: Bloomsbury, 2013. Print.

Marino, Mark C. "Critical Code Studies." *Electronic Book Review*. 4 Dec. 2006. Web. 4 Jan. 2013.

Mayer, Richard E., Ed. *Teaching and Learning Computer Programming: Multiple Research Perspectives*. Hillsdale, N.J: L. Erlbaum Associates, 1988. Print.

McGann, Jerome J. *Radiant Textuality: Literature After the World Wide Web*. New York: Palgrave, 2001. Print.

McLuhan, Marshall. *Understanding Media : The Extensions of Man.* New York: McGraw-Hill Book Company, 1964. Print.

Misa, Thomas J. *Leonardo to the Internet: Technology and Culture from the Renaissance to the Present*. Baltimore: Johns Hopkins University Press, 2004. Print.

Mitcham, Carl. *Thinking through Technology: The Path between Engineering and Philosophy*. Chicago: University of Chicago Press, 1994. Print.

Modern Language Association of America. *MLA Style Manual and Guide to Scholarly Publishing*. 3rd ed. New York: The Modern Language Association of America, 2008. Print.

Montfort, Nick and Ian Bogost. *Racing the Beam: The Atari Video Computer System*. Cambridge, Mass: MIT Press, 2009. Print.

Montfort, Nick et al. *10 PRINT CHR$(205.5+RND(1)); : GOTO 10*. Cambridge: MIT Press, 2013. Print.

Morningstar and Farmer. “Lucasfilm's Habitat.” *The New Media Reader.* Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 664-676. Print.

Murray, Janet Horowitz. *Hamlet on the Holodeck : The Future of Narrative in Cyberspace.* New York: Free Press, 1997. Print.

Neel, Jasper. *Plato, Derrida, and Writing*. Carbondale, IL: Southern Illinois University Press, 1988. Print.

Negroponte, Nicholas. “Soft Architecture Machines.” *The New Media Reader.* Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 354-363. Print.

Norman, Donald A. *The Design of Everyday Things*. United States: Basic Books. 2002. Print.

O'Gorman, Marcel. *E-crit: Digital Media, Critical Theory and the Humanities*. Buffalo, NY: University of Toronto Press, 2006. Print.

Ong, Walter J. *Orality and Literacy: The Technologizing of the Word*. New York: Routledge, 1982. Print.

Oram, Andy and Greg Wilson, eds. *Beautiful Code: Leading Programmers Explain How They Think*. Sebastopol, CA: O'Reilly, 2007. Print.

Papert, Seymour. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books, 1980. Print.

Postman, Neil. *Technopoly: The Surrender of Culture to Technology*. New York: Vintage Books, 1993. Print.

Ramsay, Stephen. *Reading Machines: Toward an Algorithmic Criticism*. Urbana: University of Illinois Press, 2011. Print.

Raymond, Eric S. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sepastopol, CA: O'Reilly, 1999. Print.

Reddell, Trace. "The Social Pulse Of Telharmonics: Functions Of Networked Sound And Interactive Webcasting." *Cybersounds: Essays on Virtual Music Culture.* 209-238. Web. 4 Jan. 2013.

Rice, Jeff. *The Rhetoric of Cool: Composition Studies and New Media*. Carbondale: Southern Illinois University Press, 2007. Print.

Rice, Jeff and Marchel O'Gorman. “Introduction.” *New Media / New Methods: The Academic Turn from Literacy to Electracy*. Eds. Jeff Rice and Marchel O'Gorman. West Lafayette, IN: Parlor, 2008. 1-7. Print.

Romanyshyn, Robert D. “The Despotic Eye and Its Shadow: Media Image in the Age of Literacy.” *Modernity and the Hegemony of Vision*. Ed. David Michael Levin. Berkeley: University of California, 1993. 339-359. Print.

Rosenberg, Scott. *Dreaming in Code: Two Dozen Programmers, Three Years, 4,732 Bugs, and One Quest for Transcendent Software*. New York: Crown Publishers, 2007. Print.

Rushkoff, Douglas. *Program or Be Programmed: Ten Commands for a Digital Age*. Berkeley, CA: Soft Skull Press, 2010. Print.

Shasha, Dennis and Cathy Lazere. *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists*. New York: Copernicus, 1995. Print.

Shneiderman, Ben. “Direct Manipulation: A Step Beyond Programming Languages.” *The New Media Reader*. Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 486-497. Print.

Simon, Herbert A. *The Shape of Automation for Men and Management*. New York: Harper Torchbooks, 1965. Print.

Smith, Brian Cantwell. *On the Origin of Objects*. Cambridge, MA: MIT Press, 1996. Print.

Spinuzzi, Clay. *Network: Theorizing Knowledge Work in Telecommunications*. New York: Cambridge University Press, 2008. Print.

Stallman, Richard M. *Free Software Free Society: Selected Essays of Richard M. Stallman*. Boston: GNU Press, 2002. Print.

Stephenson, Neal. *In the Beginning . . . was the Command Line*. New York: Harper Collins, 1999. Print.

Sterne, Jonathan. *The Audible Past: Cultural Origins of Sound Reproduction*. Durham: Duke University Press, 2003. Print.

Stroustrup, Bjarne. *The Design and Evolution of C++*. Reading, MA: Addison-Wesley Publishing Company, 1994. Print.

Suchman, Lucy A. *Plans and Situated Actions: The Problem of Human-Machine Communication*. 2nd ed. Cambridge: Cambridge University Press, 1987. Print.

Takhteyev, Yuri. *Coding Places: Software Practices in a South American City*. Cambridge, MA: The MIT Press, 2012. Print.

Thomson, Iain. “Understanding Technology Ontotheologically, or: the Danger and the Promise of Heidegger, an American Perspective.” *New Waves in Philosophy of Technology*. Eds. Jan-Kyrre B. Olsen, Evan Selinger, and Søren Riis. Basingstoke, England: Palgrave Macmillan, 2009. 146-161. Print.

Thrift, Nigel. “Movement-Space: The Changing Domain of Thinking Resulting from the Development of New Kinds of Spatial Awareness.” *Economy and Society* 33.4 (November 2004): 582-604. Web. 14 Aug. 2013.

---. “Remembering the technological unconscious by foregrounding knowledges of position.” *Environment and Planning D: Society and Space* 22(1): 175-190. Web. 1 Sept. 2013.

Torvalds, Linus and David Diamond. *Just for Fun: The Story of an Accidental Revolutionary*. New York: Harper Business, 2001. Print.

Turing, Alan. “Computing Machinery and Intelligence.” *The New Media Reader*. Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 50-64. Print.

Turkle, Sherry. *Alone Together: Why We Expect More from Technology and Less from Each Other*. New York: Basic Books, 2011. Print.

---. *The Inner History of Devices*. Cambridge, MA: Massachusetts Institute of Technology, 2008. Print.

---. *Life on the Screen: Identity in the Age of the Internet*. New York: Simon and Schuster, 1995. Print.

---. *The Second Self: Computers and the Human Spirit*. New York: Simon and Schuster, 1984. Print.

Ullman, Ellen. *Close to the Machine: Technophilia and Its Discontents : a Memoir*. San Francisco: City Lights Books, 1997. Print.

Ulmer, Gregory. *Applied Grammatology*. Baltimore: Johns Hopkins University Press, 1985. Print.

---. "Florida Out Of Sorts." *New Media/New Methods: The Academic Turn from Literacy to Electracy*. Eds. Jeff Rice and Marchel O'Gorman. West Lafayette, IN: Parlor, 2008. 21-46. Print.

---. *Internet Invention: From Literacy to Electracy.* New York: Longman, 2003. Print.

Von Neumann, John. *The Computer and the Brain*. New Haven, CT: Yale University Press, 1979. Print.

---. "Theory and Organization of Complicated Automata." *Papers of John von Neumann on Computing and Computer Theory in Charles Babbage Institute Reprint Series for the History of Computing*, vol. 12. Eds. William Aspray and Arthur Banks. Cambridge, MA: The MIT Press, 1987. 435-489. Print.

---. "Theory of Natural and Artificial Automata." *Papers of John von Neumann on Computing and Computer Theory in Charles Babbage Institute Reprint Series for the History of Computing*, vol. 12. Eds. William Aspray and Arthur Banks. Cambridge, MA: The MIT Press, 1987. 391-421. Print.

Wardrip-Fruin, Noah. *Expressive Processing: Digital Fictions, Computer Games, and Software Studies*. Cambridge, MA: MIT Press, 2009. Print.

Wardrip-Fruin, Noah and Nick Montfort, Eds. *The New Media Reader*. Cambridge, Mass: MIT Press, 2003. Print.

Weinberg, Gerald. *The Psychology of Computer Programming*. New York: Van Nostrand Reinhold Company, 1971. Print.

Weizenbaum, Joseph. *Computer Power and Human Reason: From Judgment to Calculation*. San Francisco: W. H. Freeman and Company, 1976. Print.

Winner, Langdon. "Mythinformation." *The New Media Reader.* Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 588-597. Print.

Winograd, Terry. “Thinking Machines: Can There Be? Are We?” *The Boundaries of Humanity: Humans, Animals, Machines*. Ed. James J. Sheehan. Berkeley: University of California Press, 1991. 198-220. Print.

Winograd, Terry and Fernando Flores. “Using Computers: A Direction for Design.” *The NewMedia Reader.* Eds. Noah Wardrip-Fruin and Nick Montfort. Cambridge, Mass: MIT Press, 2003. 552-561. Print.

Zizek, Slavoj. *Enjoy Your Symptom! Jacques Lacan in Hollywood and Out*. New York: Routledge, 2001. Print.

---. *The Parallax View*. Cambridge, MA: The MIT Press, 2009. Print.