Patricia Kay Galloway*

Archiving Digital Objects as Maintenance: Reading a Rosetta Machine

DOI 10.1515/pdtc-2016-0024

Abstract: Archiving digital objects consists of maintenance and conservation: the job of arresting a cultural object in time, maintaining it as closely as possible in the state in which it was created. Hardware and software provide the context in which digital objects are created, and other hardware and software provide the context in which they must be maintained, but practitioners of digital preservation are only now beginning to move seriously into the area of deciding how to perform digital objects for users. In this paper I discuss a personal effort at stopping time for hardware and what it has taught me about approaching construction of preservable platforms that can replicate the context of creation for digital objects. I will also discuss what we lose when we decide to discard environment in favor of some generalized idea of content.

Keywords: Archiving, Preservation, Digital Objects, Context of Creation

1 Introduction

Ian Bogost ("Programmers") recently drew a clear distinction between infrastructural engineering as it is known in the physical world, with its strict specifications and ethical concerns for the public interest, and what is casually referred to as engineering in the current virtual world of software; he maintained that the aspirations of the original concept of software engineering as conceived by people like Fred Brooks were (and remain, where clouds are supported by big iron) infrastructural, while the agile programming work that underpins the world of software directed at consumers, for whom it is entirely black-boxed, depends on the constant granular update. In many respects this latter phenomenon has made us forget how and why the digital environment is constructed and how its constraints may shape the affordances of the digital objects created within it.

It is therefore important to move past and below the black box to document the contexts of creation and use of digital objects, and since 2013 the students in my Digital Archiving class have begun to create virtual machines to support digital objects so as to capture and preserve the contexts from which they came—where we know what those contexts are. The problem that arises, particularly for current digital objects but really for most digital objects that have been created since the advent of the personal computer, is which software environment version(s) to choose to document, where we must guess at the context of creation, since almost every item in the originating software/hardware stack has evolved at a different rate. Digital archivists are now obliged to craft a practice of maintenance that resembles standard testing and even standard setting, since it is presently impossible to document every version of every item in the stack.

Accordingly, after 15 years of experience with the evolving practices of digital archiving in that class, we are experimenting with finding out what it takes to create a small set of virtualized canonical hardware/software systems based on our understanding of popularity and normal configuration of personal computers. We want to be able to experiment with digital objects created on legacy systems by installing on actual legacy systems the software environments that our study of the objects suggests were present, creating born-digital objects in those environments, and then creating virtual environments with the same specifications that will emulate those contexts of creation, to test how well we understand them. This approach is necessary because so much of the commercial software stack falls under intellectual property constraints so that we cannot have access to the actual code, only the executables and the physical host, if we can find a hardware example. Although efforts are now being made to preserve executables and even code, many commercial software houses are unlikely to release their code any time soon, if ever.

Why should we bother with this? If we can as archivists guarantee that the content has not been changed, why care about presentation? We care, or should do, because we can much better serve literary scholars, historians of computing, science and technology studies researchers, and even computer science theorists, if we can help them address issues like "media materiality" (Kirschenbaum, *Mechanisms*; Drucker, "Performative Materi-

***Corresponding author: Patricia Kay Galloway**, School of Information, University of Texas at Austin, 1616 Guadalupe St., Austin, TX 78701, E-Mail: Galloway@ischool.utexas.edu

ality") and "embodied interaction" (Dourish, *Action*) as well as the histories of publication forms and versioning relationships. Further, there are new demands for the retention of research software together with the datasets they have analyzed. Clearly the users of Pinterest are not much concerned with these issues, but clearly scholars are. Just because there are new affordances does not mean we should ignore them or turn our backs on the differences between using quill pens, typewriters, and computer-hosted word processors.

2 Background

This paper focuses on my conviction that anything less than the preservation of all properties of digital objects as significant ignores a large part of digital culture, which includes not only the products of work carried out with the use of computers, but the computing environment itself. The May 2016 GAO Report GAO-16-468, prepared for Congressional Requesters, "Information Technology: Federal Agencies Need to Address Aging Legacy Systems," revealed that many infrastructural information systems within the Federal government are allegedly dangerous because they are "legacy systems." One such system hit the news when it was revealed that the Department of Defense's Strategic Automated Command and Control System not only "Coordinates the operational functions of the United States' nuclear forces," but (shock-horror) "runs on an IBM Series/1 Computer—a 1976 computing system—and uses 8-inch floppy disks" (there is even a photograph of the disk, with measurement, in the report). The comment failed to point out that the Series/1 was an economical minicomputer that could not possibly have managed SACCS all by itself. Indeed the 8-inch floppies in question were probably used to boot the machine (they held a whopping 1.2 or 2.4 megabytes), not generally to handle data. The report stated that the DoD planned to update "data storage solutions, port expansion processors, portable terminals, and desktop terminals by the end of fiscal year 2017" but did not mention the computer software and hardware itself. The most striking fact that only security experts noted was that this system is still running: when it was created and ever since, it has been supported by IBM-style mainframe engineering rather than overnight-written clusters of apps—and has not been connected to the Internet (Perez, "GAO"). Finding people who understand this kind of environment, the people who carry and constitute what Peter Naur referred to as the "theory of the program" (Naur, "Programming"), is becoming even more difficult as years pass, and the reciprocal understanding between these people and the computer is lost. In enthusiasm for the new, the importance of the embeddedness of hardware/software systems in historical process is seldom taught or learned; it is forgotten along with the computer, the program, and the environment.

What I learned from my students in 2016 as we grappled with six "legacy systems" (three servers and three personal computers dating from the 1980s) is the degree to which, never having worked with non-blackboxed computers, most had never learned just how complex the contexts of digital file creation are: almost all students in the class indicated that their most powerful takeaway from it was that working with any legacy system, including one that is not very old, is a task that requires serious problem-solving just to understand the hardware/software environment and how it shapes the products created in it. I was gratified that that message was conveyed and even more gratified when two students signed up to do more such work on their projects as individual studies. I have to say that I learned a lot too, since I was forced to learn about early Macintosh machines so as to support the students.

3 A Potted Computer History

Let me take a moment to explain this large, past territory (drawn from Campbell-Kelly and Aspray, *Computer*; and Cerruzi, *History*). First there were mainframes: large computers first used to calculate artillery trajectories for World War II and then tamed into the business environment (by IBM, Control Data Corporation, Sperry-Univac, etc.) while displacing in the 1950s and 1960s female "computers" and librarians (as in the film *Desk Set* from a 1955 play; see Ensmenger, *Computer Boys*). As described in the well-known early bible of software engineering, Fred Brooks's *Mythical Man-Month*, drawn from Brooks's experience in developing the IBM 360 operating system, these mainframe computers were developed and programmed through a complex discipline of work practices that assembled teams of designers, programmers, and testers. And the operating systems that ran mainframe computers and the major applications used on them were also complexly engineered and lasted a long time. Hardware and software existed in families and instant innovation was not appreciated, especially in a world that lacked computer experts. There were few environments and only two basic encodings for binary systems: EBCIDIC and ASCII. Perhaps most important, these computers and their products were rarely seen by most people before the 1970s,

since the data they processed were mostly retained by the organization that collected or produced them, with the exception of scientific research that began to use such machines.

Next in the 1970s came minicomputers (including, the IBM Series/1 running the U. S. nuclear program), which represented in some ways scaled-down mainframes that were shared in the familial operating environments of their larger elders—innovation came primarily from new entrants into the field (Digital Equipment, Data General) and their new operating systems designed to perform more efficiently on far leaner hardware. From this iteration came the UNIX operating system developed independently by Bell Labs and released to universities in the mid-1970s. But again, even minicomputers were mostly ensconced in business, government, and carrying out research in the new discipline of computer science. The outputs of these contexts of creation were still mostly contained in-house. Further, these outputs were seldom text, but instead products of computations that were judged to be wholly constrained by their programming environment. And they generally appeared in the form of printouts on sprocketed paper.

Then, along with the increasing miniaturization of the integrated circuit, came the "computer on a chip" that made possible the microcomputer/personal computer in the mid-late 1970s, making individual ownership of computers and their encroachment into text, image, and sound outputs possible for the first time. Some were sold as kits and a large audience of tinkerers with experience in electronics bought parts and built their own, since the first microcomputers sold as stand-alone units were rather expensive, costing in the thousands. The potential market, however, looked to be enormous, and many new companies were formed, since all the parts and pieces that made up these systems were also new (memory chips shrank in size regularly and the older and now derided 8" diskettes shrank too: down to 5.25" and then to 3.5"). Many small companies sought to play the mainframe game by capturing audiences with proprietary systems as well as proprietary diskette formats (now a plague to digital preservationists). This was a period of great diversity but also of great creativity, as storage, speed, and a wide range of possible needs were explored. After some years, however, the behemoth of IBM partnered with infant Microsoft to establish one "standard" operating environment, while Apple went it alone in 1977 by selling into the home and especially educational markets and created a second such environment. Microsoft eventually dominated the office suite software market and Apple, at first subject to having a non-Apple operating system run on its machines by the actions of those dratted tinkerers, eventually developed its own graphical user interface ("borrowed" from the Xerox Alto workstation) and in 1984 Macintosh established the GUI/mouse as a standard also for what would become Windows in 1985, as Microsoft did its own borrowing.

How did computers communicate? They did not much need to at first, since mainframes were doing work in-house and terminals were attached directly to them or to newly-devised in-house networks. As businesses with multiple branches got into the act, companies rented their own phone lines. Universities and private groups also used the telephone lines to create bulletin-board systems and email lists that individuals could dial up to access information provided by special-interest groups and libraries. However, most people with personal computers used "sneaker-net," wherein they shared content (and viruses) via floppy disks. Businesses like AOL emerged to offer organized dial-up services, but to communicate initially, you and your partner had to subscribe to the same service (much like social media today). Also in the early 1970s the ARPANET was created by the American military and used both for its own purposes and by universities with grants to do military-associated work; it took nearly twenty years for its offspring to grow into the Internet as we know it and to be opened in 1993 for the commercial use that has seen an explosion as online businesses have realized that they can monetize traffic alone and give services for "free."

In the meantime, capacities for personal computers grew through the 1980s as memory got cheaper: one by one digital media emerged: photography, music, video—all of it taking up vastly more digital storage space than numbers and letters had done. Wireless connections introduced for telephones were appropriated for the Internet and especially for portable computers, thus forcing consistency. Portability also began to call for wirelessly-available storage reachable from anywhere in the form of the cloud. People adopted this apparently endless free storage for their own creations and decided that it made more sense to rent the creations of others than to buy them, via the settlement around music copyright. And finally, most individuals got the form of computer they had really wanted all along: smartphones and other forms of screens (Gruman, "Your PC"). Hence communication for whatever purpose—business or pleasure—is increasingly performed in obedience to an ever narrower set of standards that nevertheless participate in a constant churn of versions. Yet many people were not particularly interested in constant change (or could not afford it) and remained with what worked for them. More recently, we see a

movement toward understanding what makes electronic objects "ensouled" and worthy of long life in order to make the use of them more sustainable (Blevis and Stolterman, "Ensoulment"; Odom et al., "Why We Preserve"; Gegenbauer and Huang, "Longer-Lived Electronics") and have seen researchers pursue the goal of "slow technologies" to a similar purpose (Hallnäs and Redström, "Slow Technology"; Odom et al., "Critical Reflection"). We should remember that during this same time many people had no computer at all except possibly at work and were mostly entertained by a massively burgeoning cable television market: the first computer they acquired was a smartphone.

4 Case History

I came into the world of computers toward the end of the mainframe era: my first computing experience was on an IBM 370/168 time-sharing mainframe system at the University of Bonn; my first minicomputer experience was on a DEC PDP-8 running just-released UNIX in the University of London; and the first microcomputer I used was a Zilog Z-80 development system with only a monitor program and an assembly-language compiler. By capitalizing on the triumph of the integrated circuit, I built information technology for the Mississippi Department of Archives and History over twenty years beginning in 1979, first with a single Vector Graphic microcomputer running CP/M, then with an AT&T minicomputer running UNIX, and finally with networked PCs running Windows and servers running Linux. In 2000 I came to the University of Texas to teach aspiring archivists what I had learned, bringing with me my old (1991) Dell 425E desktop machine running Windows 3.1. I had been using it for around twelve years (though I had upgraded the OS once), written several books on it, and updated it using a Dell utility to cure Y2K ills. Like George R. R. Martin, I was comfortable with my writing environment (Microsoft Word 6.0, for DOS because it ran faster). Besides which, at UT I had an up-to-date machine in my office and did most of my teaching and managerial work there. At home I accessed my private email account via a dial-up modem and was still able to transport copies of ongoing writing to work when necessary on 3.5" floppies or by sending a copy to my work email account. But parts of the machine were beginning not to work very well, so I had moved most of my work to floppies or to my machine at work.

When the black and white display became unusable due to damage to the VGA controller card, I decided to purchase a laptop to use as my primary machine at home, but being short of money at the time I obtained a refurbished Dell Latitude machine originally made in 1998 and running Windows 98. I bought a discounted Microsoft Office 2000 package through the university, acquired a PCMCIA card bearing a dial-up modem, began using Netscape on the new machine, and I was ready to continue writing after transferring many 3.5" floppy backup disks to the new machine. And so I remained for some ten years (while my computers at work evolved with the times), until elements of the Latitude and its software environment began to fail. What follows shows what could be done by someone well aware of the range of systems I had experienced (my desktop machine at work at that time ran Windows 7 and had a high-speed connection to the Internet and a range of local servers), but determined to maintain an accustomed working environment.

4.1 Software

Interfaces with the outer world failed first: not because the Netscape suite I used to reach the Internet did not function on my machine, but because it was not supported widely enough on the Internet any more. Netscape Messenger had in fact ceased to be supported by my ISP for email at all, so I switched to the Outlook mail client that came with my Microsoft OS. Then for many years the substitute of Internet Explorer 6 for Netscape Navigator worked well as my browser, until in about 2015 the Web rose up and decided to kill it because it was no longer supported by Microsoft and was considered insecure. First, websites sent me nasty messages about upgrading to a decent browser; then more and more websites would not render (if I needed the information rather than the images, I found I could use my ancient Netscape Navigator as a screen reader for text—and by the way, still can). Finally, even Google refused to work with IE6, leaving me with Bing as my only search engine. How to solve this? Current browsers provide the ability to alter the user agent string that one's browser proffers to websites to tell them how to render, but my IE6 did not, and I had no idea how to make it do so, nor was I sure whether the problem was solely a software issue. At work I used Firefox as my browser, and there was an early version still available that would run on Win98. In April of 2016 I downloaded it and installed it, so I was once again back on the Internet in a way that was adequate to my needs. I could even reach my work email through webmail. Ironically, the Firefox version was even older than IE6 (and used a different layout engine), but the user agent string was equally acceptable to most websites, which had long been constructed to render in response to

the dominant browser families, thanks to the browser wars of the 1990s.

Next my private email (an AT&T account that I had used since the 1990s, upgraded with the Outlook client I received with Windows), began to refuse to download any files—not just large ones—so that I could no longer send myself files as email attachments. I had wrestled with Outlook settings for years (my Yahoo ISP was notorious for timing out), so I decided to try something else. At work I use Thunderbird: why not try that? So I downloaded the version that would work with Win98 (2.0.0.20) and once I worked out how to use this very early version this problem was also solved. I had done it by shifting to software that was at once functional, free, and reasonably easy to configure (not necessarily a trivial task). The process demonstrated how the "outer world" of the Internet is still underdetermined enough to permit such a move. And I had enough knowledge to find software that could solve my problems of externalities in the environment that I still found useful. It is true that the most elaborate commercial websites would not always render perfectly, but I used few such websites anyway. The fact is that when commercial entities failed to order users to adopt particular browsers for which they had designed their sites, web designers were forced by commercial concerns to make efforts to render their content on multiple browsers.

4.2 Hardware

Hardware was a more complex issue but at least at first not so threatening. (See **Figure 1** for rear connections on the Latitude CPi-A). First I lost the single Universal Serial Bus (USB) 1.1 port on my Latitude, which simply ceased to function. Somewhat earlier, my first 256Mb USB key (purchased for $60+ in 2004) had also died. The Latitude had a 3.5" floppy disk in an expansion bay which could also support a CD drive. As long as the files were small, I could continue to carry my files to work handily on a 3.5" diskette (amusing my colleagues) or if larger, in CD-RW form. I did not consider having the USB port fixed, since it was installed on the motherboard and I considered that major surgery. I did purchase a PCMCIA card with two USB ports on it, but my modem also needed that slot so constant swapping was not useful either. I should note that at this same time I was having some of the software troubles mentioned above, so for a time it was not possible to just use the Internet to transfer files (although once the email connection was working this became a solution again—and, I might add, one that most modern machines also use as they take to the cloud).

I continued on with my 3.5" diskettes as well until one day, having swapped the drive out of the drive bay in favor of my CD-RW drive for a bigger file, I found that when I tried to restore the 3.5" drive into the drive bay it failed and refused to be installed. In many hours spent with the Latitude maintenance manual, I discovered a new aspect of the Latitude that began to turn me to the process of "reading" this machine, of starting to think of all the affordances it had been built to support and that I had never used. It was possible, for those who needed to run both a CD and a 3.5" diskette drive at the same time, to connect the 3.5" diskette through the Latitude's parallel port (ordinarily used to interface with a printer) by using a special cable: this had been designed in from the start, at least partly to accommodate a reinstall of the OS from a CD after booting from a diskette. So I sent for such a cable (there are plenty available because apparently although one came with the new Latitude CPi-A, nobody ever used them), but alas the drive I had did not work with it either. I therefore invested in another 3.5" Latitude drive designed for its drive bay, and when it came I discovered that it apparently had been designed for a later Latitude (or Inspiron) and was not quite shaped the same. But Dell had held their laptops to design constraints that would permit interchangeable parts, and it fit into the bay, so I placed it in and found that it did work. So although I still did not have a constantly-working USB port, I did have software to connect to the Internet and both 3.5" floppy and CD-RW drives that worked. I could continue doing my work on my old machine and could use email (or Googledocs) as my temporary cloud, as many of my students had done for years and were continuing to do.

4.3 Interpellation

Why in the world did I not replace the computer or the operating system? Why did I not add a docking station or port replicator? The OS is easy: although apparently Windows XP will work on my Latitude as presently configured, and even though it is so far my favorite version of Windows, it would run very slowly without a major hardware upgrade. Further, port replicators are now rather hard to find in working order and take up desk space. But most of all, I like the bare-bones environment as it is: it does not get in the way of my writing. And it was this tiny fact that I undertook to think about, having the data points that 1) Jaron Lanier kept using WordStar for many years into the end of the 1990s just to see when it would quit (personal email); 2) George R. R. Martin still uses WordStar to write his *Game of Thrones* epic (Kirschenbaum, *Track*

Changes, 1–3); and 3) John Law has been mulling over how we are attracted to environments and ideas in the first place using Althusser's notion of "interpellation" (Law, "Machinic Pleasures" and "On the Subject of the Object") and modifying it into a notion of mutual definition between person and object. Why it took me so long to reflect personally on this issue can only be explained by the fact that getting really reflexive is very hard to do until we find ourselves truly discomfited by being forced from comfortable habit, especially when our machinic surroundings have been so blackboxed that even after long use we are nevertheless seldom well enough trained to cope with repairing or refitting them. Even if one had the skills, there would be the problem of tools: repair shops are not necessarily equipped with the tools used for taking older computers apart and they are abstruse enough that it is hard to find them.

Until I started struggling with my own home computer, I forgot what I had learned from my earliest days of personal computers—which were far simpler machines—about the confines that a computer constitutes. I forgot that I had made two computers exchange files in spite of incompatible floppy disk formats in the 1980s by building a null modem cable to allow direct exchange of files between the machines, long before the Internet or even convenient online bulletin boards. In fact I had been seduced by the ease of network communication infrastructures and half-standardized tools and supported by the relative uniformity of the Internet. Yet I was able to create a working environment anyway, in spite of the failure of "anybrowser" efforts to persuade web designers to accommodate all browsers ever made for the sake of accessibility and my disinclination to spoof the user-agent string by which browsers announce themselves. This is possible because the network environment, to be profitable, cannot entirely forbid somewhat aging technology, and (especially) nobody is being paid to eradicate the remaining open possibilities. On the other hand, the evolution of the network is constantly moving toward network-hosted computer work and provisioning of software as well as storage (so that we rent software as we do music), although it requires high-speed connections that are harder for older equipment to support. George R. R. Martin takes the easy way out: he writes on his WordStar machine and uses a modern computer to surf the Internet. I found that I had arrived at the same solution, in my case between home and work.

Growing up with the Depression-wrought ethos of my father's construction of an amateur radio system that remained functional with little change for more than thirty years—he drew on war-surplus parts and his and his brother's self-taught and military-taught electronics skills—it never occurred to me that I should *not* keep my old computers and my comfortable working environments running for as long as they would. As I have begun to collect older machines for the Digital Archaeology lab that I use for class and individual student projects at the University of Texas School of Information, I find that many people have hung on to old equipment that may not even work but has certainly been used for longer than the seller intended (Newitz, "My Laptop"; Turkle, *Second Self*). My example shows that what is forcing people to adopt "upgrades" is frequently not choice, but a result of the loss of support for proprietary software and hardware and the necessity to negotiate a steep learning curve in order to get around the problems that arise from continuing to use an accustomed environment. Working with my Latitude in the way that I did, exhausting many of its capacities in order to confront the shifting platform infrastructure of Internet communication—in fact working on the Latitude alone as a closed system, for writing, there was no need to make any changes at all until I needed to export a file—I began to construct an idea of just how important this view of computer hardware persistence is, not only to sustainability in the design of electronics, but to the necessity of being able to understand and replicate temporally-situated environments in order to preserve a reasonable semblance of past platforms and the relationships of users and tools that uniquely created those contexts. This is the very problem that archivists face in providing a true performance of digital objects created and viewed in older environments.

Michael Mahoney, the late dean of computer historians, frequently observed that since technologists and programmers do not often express themselves on how they have built things, it is important to look at what they have made for clues to why and how they solved those problems. Similarly, Thomas Haigh has remarked that important strides in the Science and Technology Studies (STS) program have been made by connecting blackboxed machines with the social contexts from which they have come (Haigh, "Unexpected Connections"). In 2003 Mahoney wrote an often-quoted paper entitled "Reading a Machine." In it he chose to use the example of the Model T Ford with its interchangeable parts made conveniently available to owners through ordinary hardware stores in order to examine the political economy that Ford built underneath the machine itself, which was itself "readable" from the machine (Mahoney, "Reading")—we have just seen the repetition of the practice in the persistence of physical features on Dell laptops. Bruno Latour and Steve Woolgar showed, as a result of their participant observa-

tion in a scientific laboratory (*Laboratory Life*), the way that science itself advances by blackboxing experimental apparatus such that advancement by others depends upon the adoption of the same apparatus (and assumptions) dictated by pioneers in a field—and this exercise has much to teach about how commoditization works in technological "development" and standardization works in establishing an infrastructure. Andrew Pickering's work further explored the phenomenon of blackboxing by showing that said apparatus came to seem to exhibit what he called "material agency" in that specific theory-laden machines (like electron microscopes) came to seem not to require any explanation or contestation of their underlying assumptions as they became naturalized (*Mangle of Practice*). Finally, John Law has contemplated the whole notion of why STS scholars want to study machinic systems in the first place and how they might use their own situatedness to do so, by taking inspiration from Althusser's theory of interpellations to unpick from varying directions the details and mutual performances of subject-object relationships between people and machines (Law, "Subject of the Object," "Machinic Pleasures," and *Aircraft Stories*).

5 Reading Machines (and Decentering the Object?)

All of the foregoing was necessary for me to present the historical development into which my tinkering with the Latitude CPi-A fits, because my very modest tinkering was all about getting out of the box—once a loose set of new "standards" had spottily fought with the setup that my particular comfortable work environment represented. This case situated my own struggles into the greatly accelerated development history that would lead up to where we are today. We now see a shift in importance from internal consistency to external consistency and a vast narrowing of connection possibilities, down to the paucity of the very early days of personal computers, when connection was rare and mostly proprietary, and today's environment of mainframes and their replacements, servers.

In 1998 when my Latitude machine was manufactured, several big changes were on the horizon. Many people have heard of Doug Reside's description of the "Rosetta Machine," which is ideal for the capture of a lot of vintage data from older machines because it has many connectors to all kinds of peripheral machines ("Rosetta Computers"). His version of this machine is the Apple PowerBook G3 made in 1998 and known affectionately as "Wallstreet." This machine was not cheap and was very heavy, but it had on board not only multiple interfaces (According to Charles Moore, "arguably the most comprehensively complete and expandable PowerBook ever built, with its full set of classic PowerBook ports, two PC Card slots [allowing upgrades], ethernet and IR connectivity") and also two drive bays, one for a CD and the other for a 3.5" diskette, both capable of supporting an additional battery or additional removable media drive options. And because there were two PC card slots, many other external peripherals could be interfaced. Clearly this amazing machine was widely admired at the time—not only by present-day digital archivists. The first Wallstreet was introduced in May 1998. According to Charles Moore's "Compleat Guide to Wallstreet PowerBooks," the June 1998 issue of *Macworld*'s article on the machine by Henry Bortman suggested that this machine would threaten the market for desktop machines; and it is amazing and rather odd that such a laptop machine would have so many and such a variety of external connections to peripheral machines when its ostensible use was that it could be used on the road.

I would argue, however, that if we read this machine as Mahoney suggests, what we see is a machine at a crossroads, or perhaps as Law would argue, the nexus of many possible machines or machine-intentions, at a point where although the business world was not yet beginning to shift from a variety of connection methods to a few, the success of PC-clone laptop machines was forcing Apple into support for fewer proprietary connections and more standard ones. This made for a moment of rich variety in the endless chain of ever-changing planned obsolescence, which emerged when there was a moment of uncertainty in the combination of market and technology where some aspect of constructed user needs (speed, storage space—yet still a need for a variety of interfaces) was pushing against the limitations of the platform. At such points, where the futures of communications and storage across the boundary of the individual machine are uncertain, machines might be built that could interface to practically anything.

This moment is interestingly illustrated if we compare my humble Latitude (several pounds lighter and having no proprietary connections) to the contemporary Wallstreet. In November 1997, Charles Piller reported in the *Los Angeles Times* that Steve Jobs had announced that Apple would begin to sell directly to customers online, just like Dell and Gateway were doing—but Piller observed that in most cases comparable machines would cost less: in the case of a Latitude CP versus a Wallstreet, $934 less for the Latitude (Piller, "Different Thinking").

And interestingly, my lowly Latitude had something to teach about this moment of change in the industry: along with the regular I/O connectors Dell had introduced something new: a USB1.1 connector, not present on the Wallstreet I (see **Figure 1**).

Figure 1: Rear connections for Dell Latitude CPi-A (1998).

That this became important is easily seen on examination of the Wallstreet "Pismo," brought out in late 1999. This machine was not made at Apple's factory in Ireland, but was instead produced in Taiwan, doubtless anticipating a larger penetration of the market for somewhat lighter but still capable laptops. Significantly, the Pismo replaced for the first time the long-preferred Apple SCSI interface with *both* two USB connectors and two Firewire connectors (see **Figure 2**),

Figure 2: Rear connections for Apple Powerbook G3 ("Wallstreet," 1998) on the bottom; Apple PowerBook G3 ("Pismo," 1999) on the top.

although a single PC card could bring the SCSI back, and it should also be noted that Apple engineers were significant contributors to the Firewire standard. Perhaps in response a later Latitude, the 2003 C840—which still shared the form factor of drive bays with my CPi-A—was itself equipped with two USB ports as well as acquiring a Firewire port (see **Figure 3**; **Table 1** provides a comparison of connections on all four machines).

Figure 3: Rear connections for Dell Latitude C840 (2003).

Table 1: Connectors Present on Late 1990s to Early 2000s Laptops

Computer																OS
Lat Cpi-A				serial port	parallel port				VGA monitor	Infra-red	1 USB 1.1		1 PC Card	RJ-11	CD-ROM or 3.5" flop	PS-2	Win 98
Wall-street	audio out	audio in	AD B	serial port		Ether net	HD-30 SCSI	TV out	VGA monitor	Infra-red			2 PC Cards	56K modem	CD-ROM		OS X Jaguar
Pismo	audio out					Ether net					2 USB 1.1	2 Fire Wire	1 PC Card	56K modem	DVD-ROM		OS X Tiger
Lat C840	audio out	audio in		serial port	parallel port	Ether net		S-Video	VGA monitor	Infra-red	2 USB 1.1	1 Fire Wire 4-pin	2 PC Cards	56K modem RJ-11	DVD-ROM and 3.5" flop	PS-2	Win XP

What is amazing is that today's laptop has none of these ports except multiple USBs and wireless capabilities. One thing that drove this trend was the USB flash drive, which suddenly offered a convenient way to carry a large amount of data from one place to another; the other thing was a shift in business plans that I have already mentioned: from internal consistency (what was convenient for the manufacturer of the computer, depending on what standardized peripherals were available) to external consistency (from the moment of the debut of the USB port, what was convenient for multiple manufacturers of peripherals, which was standardization of connectors to a few as possible). According to Andrew Cunningham, this phenomenon has simplified things for the user at the possible expense of better performance (from the Firewire and later Thunderbolt technologies); he suggests that many applications may become wireless, but cannot do so

where high speed is required (Cunningham, "Brief History of USB"). So while wireless may work for smartphones, it is unlikely to host big-data computations, and at least for a while USB will probably continue to dominate. Compare the elegantly minimalist Apple MacBook Pro, whose four Thunderbolt 3 ports can perform the characteristics of USB, HDMI, VGA, or handle a separate display; there are no other ports visible on the outside of this extraordinarily thin closed laptop (see **Figure 4**).

Figure 4: Side connections for Apple MacBook Pro (2015).

6 Conclusion

The characteristics of the "Rosetta Machines," apart from their accumulations of technologies, remind us first that it took a long time to establish a worldwide communication web and second that it has also taken a very long time to get everyone attached to it, during which the personal computer has morphed to many applications, which progressively demand different kinds of connections and ownership patterns of computer-associated equipment. The market is pushing most users into the realm of smartphones and other wireless screens, software as a service, and vast cloud storage to satisfy work and entertainment needs. On the other hand, not everyone can afford even the cheapest smartphone, and some people want to control how (or if) they attach to the Internet as increasing surveillance and hacking target the network and new concerns for privacy emerge. Hence interpellation, in this case the shaping of the user by the machine, is not perfectly predictable because the user's experience is a complex conversation; every situation emerges from a range of actions that can be read, not just from the machines themselves but from those who made them and those who use them. As we have seen, each personal computer becomes an environment crafted to some extent by the user, within the restrictions of hardware and the possible connections that the user can choose, at least operating system software as well as proprietary application software, since in both cases there are configuration choices that can be made by the user. The user can, of course, always misuse both hardware (gamers' overclocking of their processors) and software (Arnold Wesker's exploitation of the space designed for the three-character file-type abbreviation to extend his filenames for more precision (Kim et al., "Automated Batch").

It seems to me that the lessons here, gleaned from my experience in making machine functions persist and the application of Mahoney's "reading" to multiple machines, suggest that it is a mistake to limit ourselves to preserving only the *output* of vintage systems, even if we preserve the ability to perform it perfectly. We need also to preserve past users' relationships with their machines, because only then can we understand how interacting with machines changes how we act on the world and other people *through* those machines. There is a clue to this in the Salman Rushdie Workstation at Emory and the collection of all his machines (with their contents) from the period when he worked while in hiding. Rushdie himself has attributed to this preservation his ability to work effectively on his memoir (Williams, "Rushdie: Digital Archive"). The machines are not just the medium: they are also actors that make us in interaction as we make them. Hence there is more to Rosetta Machines—and virtual machines reproducing them—than capturing outdated media. All machines have a bit of Rosetta in them in that they are doors to small scraps of the history of our digital culture from their position in the flow of invention and intention. If archivists are not to "curate" this history, who is? And what are we curating if not the complexities of the interpellation wrought upon the user by the original environment, a relationship that can perhaps be established again. Can we do less?

References

Blevis, Eli, and Erik Stolterman. "Ensoulment and Sustainable Interaction." *Proceedings of the International Association of Societies of Design Research*, The Hong Kong Polytechnic University, November 12–15, 2007.

Bogost, Ian. "Programmers: Stop Calling Yourselves Engineers." "Technology", *The Atlantic*, 11/5/2015, accessed 8/25/2016 at http://www.theatlantic.com/technology/archive/2015/11/programmers-should-not-call-themselves-engineers/414271 (accessed March 1, 2017).

Brooks, Frederick P. *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1975.

Campbell-Kelly, Martin, and William Aspray. *Computer: A History of the Information Machine*, 2nd ed. Boulder, CO: Westview Press, 2004.

Cerruzi, Paul E. *A History of Modern Computing*. Cambridge: MIT Press, 1998.

Cunningham, Andrew. "A Brief History of USB, What It Replaced, and What Has Failed To Replace It." "Gear & Gadgets", *Arstechnica*, 8/17/2014, accessed 8/25/2016 at http://arstechnica.com/gadgets/2014/08/a-brief-history-of-usb-what-it-replaced-and-what-has-failed-to-replace-it (accessed March 1, 2017).

Dourish, Paul. *Where the Action Is: The Foundations of Embodied Interaction*. Cambridge: MIT Press, 2004.

Drucker, Johanna. "Performative Materiality and Theoretical Approaches to Interface," *DHQ: Digital Humanities Quarterly* 7.1 (2013). At http://www.digitalhumanities.org/dhq/vol/7/1/000143/000143.html (accessed March 1, 2017).

Ensmenger, Nathan. *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. Cambridge: MIT Press, 2010.

Gegenbauer, Silke, and Elaine M. Huang. "Inspiring the Design of Longer-Lived Electronics through an Understanding of Personal Attachment." *Proceedings of DIS 2012*, 635–644. ACM.

Gruman, Galen. "Your PC Is Simply Another Mobile Device." *Smart User*, Infoworld, 9/20/2016, accessed 9/20/2016.

Hallnäs, Lars, and Johan Redström. "Slow Technology–Designing for Reflection." *Personal and Ubiquitous Computing* 5.2 (2001): 161–91.

Haigh, Thomas. "Unexpected Connections, Powerful Precedents, and Big Questions: The Work of Michael S. Mahoney on the History of Computing," in Thomas Haigh, ed. *The Histories of Computing: Writings of Michael Sean Mahoney*. Cambridge: Harvard University Press, 2011.

Kim, Sarah, Lorraine A. Dong, and Megan Durden. "Automated Batch Archival Processing: Preserving Arnold Wesker's Digital Manuscripts." *Archival Issues* 30.2 (2006): 91–106.

Kirschenbaum, Matthew G. *Mechanisms: New Media and the Forensic Imagination*. Cambridge: MIT Press, 2008.

Kirschenbaum, Matthew G. *Track Changes: A Literary History of Word Processing*. Cambridge: Harvard University Press, 2016.

Latour, Bruno, and Steve Woolgar. *Laboratory Life: The Construction of Scientific Facts*. Princeton: Princeton University Press, 1979.

Law, John. "On the Subject of the Object: Narrative, Technology, and Interpellation," *Configurations* 8.1 (Winter 2000): 1–29.

Law, John. "Machinic Pleasures and Interpellations." 2001. Centre for Science Studies, Lancaster University, Lancaster LA1 4YN, UK, accessed 8/25/2016 at http://www.comp.lancs.ac.uk/sociology/papers/Law-Machinic-Pleasures-and-Interpellations.pdf (accessed March 1, 2017).

Law, John. *Aircraft Stories: Decentering the Object in Technoscience* (Durham: Duke University Press, 2002).

Mahoney, Michael S. "Reading a Machine", accessed 8/25/2016 at https://www.princeton.edu/~hos/h398/readmach/modeltfr.html (accessed March 1, 2017).

Moore, Charles W. "Low End Mac's Compleat Guide to Wallstreet PowerBooks," accessed 8/25/2016 at http://lowendmac.com/2014/low-end-macs-compleat-guide-to-wallstreet-powerbooks (accessed March 1, 2017).

Naur, Peter. "Programming As Theory Building," *Microprocessing and Microprogramming* 15.5 (May 1985): 253–61.

Newitz, Anna Lee. "My Laptop," in Sherry Turkle, ed. Evocative Objects: Things We Think With. Cambridge: MIT Press, 2007, pp. 86–91.

Odom, William, James Pierce, Erik Stolterman, and Eli Blevis. "Understanding Why We Preserve Some Things and Discard Others in the Context of Interaction Design." *Proceedings of Computer-Human Interaction*, Association for Computing Machinery, 2009, 1053–62.

Odom, William, Richard Banks, Abigail Durrant, David Kirk, and James Pierce. "Slow Technology: Critical Reflection and Future Directions." DIS 2012 Workshop, Association for Computing Machinery.

Perez, Roi. "US GAO Finds Nukes Are Controlled by Computer from 1970's," *SC Magazine UK*, May 26, 2016, accessed 1/20/2017 at https://www.scmagazineuk.com/us-gao-finds-nukes-are-controlled-by-computers-from-1970s/article/530980/ (accessed March 1, 2017).

Pickering, Andrew. *The Mangle of Practice: Time, Agency, and Science*. Chicago: University of Chicago Press, 1995.

Piller, Charles. "Different Thinking–Except on Price," *Los Angeles Times*, accessed 8/25/2016 at http://articles.latimes.com/print/1997/nov/17/business/fi-54720 (accessed March 1, 2017).

Reside, Doug. "Rosetta Computers," in Kirschenbaum, Matthew G., Richard Ovenden, and Gabriela Redwine, *Digital Forensics and Born-Digital Content in Cultural Heritage Collections*. Washington, D. C.: CLIR, December 2010, p. 20.

Turkle, Sherry. "Personal Computers with Personal Meanings," Chapter 5, *The Second Self: Computers and the Human Spirit*, 155–182. Cambridge: MIT Press, 2005 (lst ed., 1984).

U. S. Government Accounting Office. GAO Report, GAO-16–468, "Information Technology: Federal Agencies Need to Address Aging Legacy Systems." Report available at http://www.gao.gov/assets/680/677454.pdf (accessed March 1, 2017).

Williams, Kimber. "Rushdie: Digital archive at Emory 'allowed me to write' memoir." *Emory Report*, March 8, 2012. At http://news.emory.edu/stories/2012/03/er_rushdie_digital_archives (accessed March 1, 2017).

Bionote

Patricia Kay Galloway

Patricia Kay Galloway is Professor at the School of Information, University of Texas, Austin. She has published extensively. Her recent research projects include institutionalization of digital repositories and appropriate appraisal practices for digital records. She is also interested in understanding how archiving and cultural preservation fit into their historical and cultural contexts. Her "Commentary on Integrating Research and Teaching" appeared in *PDT&C,* issue on preservation education [43.1–2 (2014): 51–53].