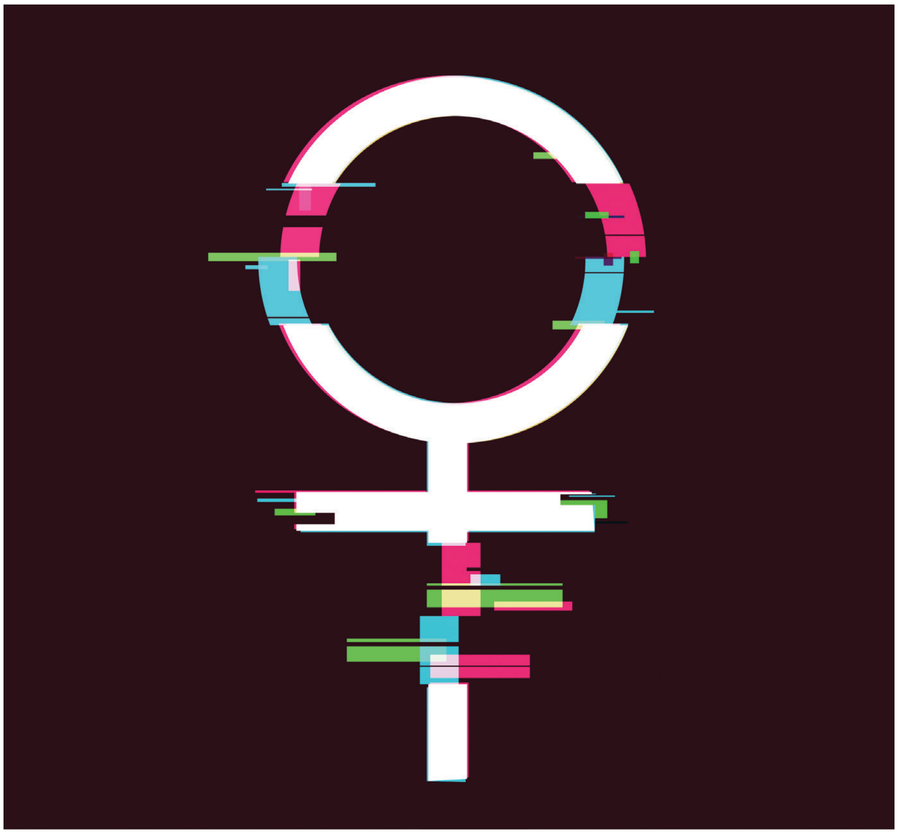
DOI:10.1145/3473610

Thomas Haigh

Historical Reflections
Women's Lives in Code

Exploring Ellen Ullman's 'Close to the Machine' and AMC's 'Halt and Catch Fire.'

In this column, I look at two vivid depictions of programming work: Ellen Ullman's *Close to the Machine*, a memoir from 1997, and the television show "Halt and Catch Fire," which ran for four seasons starting in 2014. Both have central characters whose technology careers began in the 1970s and are followed through the mid-1990s—from the glory days of minicomputers and the first personal computers to the dawn of our current online existence. Both center on women who built their identities around computer programming, sometimes to the detriment of their personal relationships.



Getting Close to the Machine

When Ullman's book first appeared the computing world it described seemed quite different from the green screen eras described by Steven Levy in *Hackers* and Tracy Kidder in *The Soul of a New Machine* (both explored in previous "Historical Reflections" columns this year: January and April). Microsoft Windows had replaced the text interfaces of CP/M and timesharing systems. Most workplaces had already computerized and powerful personal computers were increasingly common in the home. The explosive growth of the World Wide Web was transforming the Internet from an academic enclave into a bustling shopping mall. Experienced programmers, like Ullman, were in great demand as the tech world thrilled with the excitement of unfolding possibilities.

The bigger shift, though, was literary: from the external perspective of *The Soul of a New Machine*, itself a classic of literary non-fiction, to a startlingly frank first-person voice. Most discussion of women's careers in IT focuses on sexism, hostile work and study environments, and ways to overcome barriers standing in the way of more equal participation. Ullman has surprisingly little to say about these issues, but she is acutely aware that as a secular middle-aged Jew, bisexual woman, former communist, and Ivy League English graduate, she falls outside the typical demographic parameters of a software developer. This perhaps challenged her to think more deeply about her life and choices, and certainly equipped her to tie together the personal and professional with exceptional verve. Yet she is more concerned with telling us what it feels like to be a programmer, specifically a programmer who tries to make sense of her own part in the evolution of capitalism, than in documenting the special challenges faced by women in IT.

This, she shows us, is what it feels like to stay up all night trying to configure a DBMS. This is how you square your career as a contract developer working for large corporations with your past as a communist agitator. And over there, Ullman confides as she continues our backstage tour of her own head, just past a prized stack of old Unix manuals and rubbing up against the fear of aging, you will see some disturbingly algorithmic sex with a callow cypherpunk named Brian who "looks exactly the way today's computing genius is supposed to look: boyish, brilliant, and scary."

Exploring the Work of Ordinary Developers

In some ways, though, Ullman's experience is far more typical than that of the celebrated hackers Levy wrote about, or the billionaire entrepreneurs who receive most attention from technology writers. Most programmers, particularly back in the late 1970s when Ullman started out, did not have computer science degrees. In the 1990s, computer systems were generally much more important to people's work lives than to their personal lives, given the investment made by most organizations to computerize their administrative processes.

Most developers produced custom database-driven application systems for the kinds of user organizations she describes, like banks, small businesses, and non-profits. Yet writers who have looked at software development focus on commercial packages and operating systems. Ullman drops hints of her past as an early employee of Sybase (the original developer of SQL Server) and mentions receiving windfalls from options at two startups. In those jobs she must have sat alongside people who went on to buy vineyards or become famous venture capitalists. But the work she chose to relate in detail is the analysis, design, and implementation of a custom application to handle the needs of local AIDS patients.

Most programmers, particularly back in the late 1970s when Ullman started out, did not have computer science degrees.

If most writing about software mimics Ayn Rand's narrative in *The Fountainhead* of the visionary architect determined to create a monumental structure, Ullman's programming work is more like the typical experience of a commercial architect, taking pride in designs for supermarkets or low-rise apartment buildings. Some of the book's most interesting passages depict Ullman's interactions with the "end users" themselves and the managers and supervisors whose desires shaped the systems she was programming. The "fleshy existence" of these users complicates the abstract versions of their needs and behaviors she has built into the system.

The book was published by City Lights books, an imprint of the legendary San Francisco bookstore. One legacy of Ullman's immersion in radical politics, queer culture, and feminism, three things the city used to be known for, may be her unspoken conviction that her career and life deserve to be unflinchingly documented and publicly exhibited even though she did not start a famous company or invent a technology. Her determination to capture the subjective interior feelings of a character going about her ordinary business and her sense of herself as an outlier in her profession both put Ullman in a distinctively female literary tradition exemplified by pioneering modernist Virginia Woolf.

The View from Inside

In one passage Ullman contrasts the poised banalities expressed by a vice president of what appears to be Visa Inc., to whom programming seems trivial and mundane, with her own anxious fascination with the interaction of humans and machines. The manager voices faith that systematic development methodologies and careful systems analysis work will assure the success of every development effort. Months later, Ullman catches the same manager in a confession that her firm's core transaction system is an ageing mass of mainframe assembly code understood by only three programmers. "The slip-space opened before us," writes Ullman. "The world and its transactions sat on one side. The programmers, the weird strange unherdable cats, roamed freely on the other. The vice president had peered into the abyss. Then she stepped back. 'Don't tell anyone I said that,' she said."

Ullman identifies the central challenge of software development as incompatibility between the clean, formal world of computer logic and the fuzzy humanity of the workplace: "The project begins in the programmer's mind with the beauty of a crystal. I remember the feel of a system at the early stages of programming, when the knowledge I am to represent in code seems lovely in its structuredness. For a time, the world is a calm, mathematical place. Human and machine seem attuned to a cut-diamond-like state of grace.... Then something happens. As the months of coding go on, the irregularities of human thinking start to emerge. You write some code, and suddenly there are dark, unspecified areas.... Details and exceptions accumulate. Soon the beautiful crystal must be recut. This lovely edge and that one are gone. The whole graceful structure loses coherence. What began in a state of grace soon reveals itself to be a jumble. The human mind, as it turns out, is messy."

Building on this insight she brilliantly explains the apparent paradox, visible also in Levy's *Hackers*, that computer nerds can be conspicuously sloppy and disorganized in their personal life but aggressively precise in arguments. According to Ullman the "stereotype of the programmer, sitting in a dim room, growling from behind Coke cans" is rooted in their need to "talk all day to a machine that demands declarations." "The disorder of the desk, the floor; the yellow Post-It notes everywhere; the whiteboards covered with scrawl: all this is the outward manifestation of the messiness of human thought. The messiness cannot go into the program; it piles up around the programmer. Soon the programmer has no choice but to retreat into some private interior space, closer to the machine, where things can be accomplished."

Fear of Obsolescence

The tech industry prizes boyishness and novelty, making Ullman's concern with history and the passage of time a refreshing expression of humanity. She equates human aging with technological obsolescence, linking her own fears as a woman soon to enter her 50s ("a depressed, uninteresting region") with her emotional connection to the material detritus left by relentless technological change. Early in her career, Ullman turned down an invitation to apprentice to an older programmer who "had made his peace with his own obsolescence." The man was maintaining code on a platform developed in the 1950s. He offered her a chance to take his place, to one day become "the last human on earth who knows how to program in 1401 Autocoder." Ullman turned down this invitation but came to share his belief that there was "perverse dignity in knowing obsolete arcana." Judging from the hundreds of thousands of views given to retrocomputing videos on YouTube that belief is now more widely shared.

I hope for both their sakes that Brian, her youthful "anarcho-capitalist" lover, is a heavily disguised composite or an outright invention. He personifies an emerging Silicon Valley culture that intrigues and horrifies Ullman. Brian is uncultured, almost feral, and committed to a theoretical polyamory that is in practice mostly celibate. He dreams of setting up porn servers in data havens and developing cryptographical banking systems so secure that users have to balance their own accounts. If real, Brian might have been a housemate of Peter Thiel or Elon Musk, who even then were dreaming up parts of what became PayPal.

Brian's obvious unworthiness of our heroine is crystallized in a fireside chat early in their relationship. She shows him her prized, spiral-bound Bell Labs manual for Unix Release 3.0, issued in June 1980. "It came," she remembered, "from the days when I stopped being a mere programmer and was first called a 'software engineer.'" He calls it "trash" and tells her to throw it away. Her collection of obsolete manuals must, she reflects, nevertheless contain "some threads, some concepts, some themes that transcended the details, something in computing that made it worth being alive for more than 35 years."

Ullman gives a wonderful description of the sheer flood of paper and disks that engulfed the technologically committed in the 1990s: gigantic catalogs, updates to the Microsoft Professional Developer Network library, specialist magazines and journals, bulky manuals, new tools. Staying current means constantly mastering new technologies. Ullman relates with pride that she has taught herself "six higher-level programming languages, three assemblers, two data-retrieval languages, eight job-processing languages, seventeen scripting languages, ten types of macros, two object-definition languages, sixty-eight programming-library interfaces, five varieties of networks, and eight operating environments." Beginning to weary of this, she wonders if perhaps the process "is simply unnatural for someone over thirty-eight," particularly as career success tends to see developers move away from the machine and into new roles managing the people who know how to do the actual work. Yet she still feels joy when she is able to help a subcontractor find the mistake in his program. "For one more day at least," she writes, "I would still be thought of as 'technical.'" That meant a lot to her.

For me at least, the joy of discovering Ullman's book was reading for the first time a faithful description of my own experience building client-server and Web systems.

Ullman Intertwines Her Life and Work

Ullman's book grips because she puts us right inside the mind of a 1990s software developer. In a foreword to a re-release of *Close to the Machine*, Jaron Lanier recalled his amazement on first reading it, to discover "a computer nerd who could write." It formed "a bridge between reality at large and the empire of nerds, which seemed non-reactive and immune to subjectivity, beauty, love, or the acknowledgment of fundamental frailty."[a] More than 20 years on, I have encountered no comparably compelling memoirs written by other programmers.

Tracy Kidder and Steven Levy both serve as viewpoint characters for their readers, apparently normal people who closely observe obsessive hardware and software developers on our behalf. The importance of Silicon Valley and coding has been hard to ignore recently, but those with the skills and inclination to write about their experiences have usually been non-technical youngsters who stumble into the field. Consider, for example, the gulf between Ullman's perspective and the 20-something memoirist and former New York publishing assistant Anna Wiener, whose recent *Uncanny Valley* critiqued startup culture from the viewpoint of a customer service worker. Plenty of novelists have tried their hand at depicting programming and other IT work but most tend to get hung up on surface detail (I mention a few exceptions, including Ullman herself, in the "Further Reading" section at the end of this column).

Ullman's book appeared just as memoir writing was beginning to boom, with books like Frank McCourt's *Angela's Ashes* dominating best-seller lists and winning major awards. Most memoirs traded on the dramatic (and sometimes disputed) life experiences claimed by their creators: growing up in abject poverty, suffering tragic loss, re-

a Lanier's introduction appeared in the 2012 Picador edition.

covering from drug addiction, working in the sex industry, being diagnosed with terrifying diseases, fighting in wars, or building schools in Afghanistan. They offer vicarious experiences that most of us are happier to avoid encountering in person.

That is probably because coding does not provide the obvious narrative hooks of drug addiction or war. For me at least, the joy of discovering Ullman's book was reading for the first time a faithful description of my own experience building client-server and Web systems with MS Access, Oracle, SQL Server, and ColdFusion to underwrite an unusually comfortable graduate school lifestyle. When I stumbled across her book in the Center City Philadelphia branch of Borders, not far from the shelves of fat Que and O'Reilly programming manuals, I was hooked. Nobody had described so captivatingly the subjective experience of programming or the life of a freelance application developer. I gave a copy to my father and another to my dissertation advisor (an expert on late 19th-century labor), hopeful that it might communicate about new modes of work that I had not quite known how to explain myself.

The Changing Nature of Capitalism

Ullman contrasts the relentless impermanence of her own career, with its project-based alliances of convenience, virtual companies, and failed startups, with the determination of earlier forms of capitalism to present at least a façade of solidity. This is symbolized by the marble lobby of a grand bank she remembers her mother dressing up to visit, a physical space Brian hopes to replace with cryptographic algorithms.

She also remembered her father's accounting practice, built on long-term human connections, and the gamble he took to put together funding to secure her legacy: ownership of a small office building close to Wall Street. The cycle completes when she narrates a visit with her sister to meet with its struggling tenants. One complains of being unable to pay rent because of "the modems"—senior financiers are now telecommuting from the suburbs. Her career is helping to depopulate her own building. I have read a lot of books fulminating about the evils of "late capitalism," Silicon Valley, and neo-liberalism but I have found nothing in them as honest and human as Ullman's efforts to reconcile the paradoxical strands of her own life. It is the opposite of the "determined solipsism" Brian shares with Levy's hackers.

Near the end of the book, Ullman returned to product development at the request of a venture capitalist, to attempt to salvage some value from a startup that is already failing. Her description echoes Kidder's experience with Data General engineers more than 15 years earlier: "A merry kind of hysteria took over the programmers. The situation was impossible, the deadline was ridiculous, they should have been completely demoralized. But, somehow, the absurdity of it all simply released them from the reality that was so depressing the rest of the company. They played silly jokes on one another. They stayed up late to see who could finish their code first. The very impossibility of success seemed to make the process of building software only that much sweeter."

They are all of them, Ullman realizes, deeply fortunate to be engineers "who built things and took our satisfaction from humming machines and running programs." Whether the company was liquidated or not, they had succeeded in transforming mere "scratchings on a white board" into something that worked. And then she breaks up with Brian and the book ends.

"Halt and Catch Fire"

The great strength of "Halt and Catch Fire," which covers the evolution of personal computing and networking from 1983 to 1995, is its ability to capture such moments of creative flow. Even at its worst, in its sputtering first season, the show has a more deeply felt connection to the work of programmers and engineers than anything else on television. As "Halt and Catch Fire" progresses it comes to share something else with Ullman's memoir and Kidder's classic book: it affirms the value of careers that do not necessarily lead to fame, power, and great wealth. The show matures into a moving examination of the creative joys and personal sacrifices its characters find in lives built around technological creativity.

Yet early on the show almost collapsed under the weight of the narrative template that has come to dominate the stories we tell about the computer industry: egotistical men becoming billionaires by bending reality to their will. "Halt and Catch Fire" only began to work after it recentered on its female characters and redefined success. It was marketed as AMC's follow-up to its hit series, and first original drama, "Mad Men." In its first season the show attempted to do the same things as "Mad Men," but in the 1980s Texan computer industry and with bad clothes. Its title, best ignored, was explained as an "early computer command" that forced "all instructions to compete for superiority at once." (HCF was actually a jokey unofficial mnemonic for an undocumented instruction that caused early Motorola processors to cycle relentlessly, probably for diagnostic purposes).[b]

"Halt and Catch Fire"'s inexperienced creators, Chris Cantwell and Christopher C. Rogers, were likewise drawn to the idea of a computer industry drama patterned after "Mad Men." That show's protagonist—charismatic 1960s advertising executive Don Draper—had a tortured personal life and a traumatic backstory. He arrived amid a wave of shows centered on charismatic, brilliant, and psychologically complex antiheroes, a trend kicked off by "The Sopranos" and adopted by other acclaimed dramas such as AMC's own "Breaking Bad." "Mad Men" was the rare workplace drama that took work seriously. Its most resonant moments centered on obsolete technology and defunct brands.[c] The emotional manipulation of Draper's advertising pitch for the Kodak Carousel slide projector as a personal time machine fueled by nostalgia, widely viewed as the show's finest moment, was compounded by our knowledge that the users who carefully ordered their slides to tell family stories are themselves mostly memories at this point. It is also difficult to forget a mordantly humorous incident that capped a season of asides about technological

b Gerry Wheeler. Undocumented M6800 Instructions. *Byte 2*, 12 (Dec. 1977), 46–47; https://bit.ly/3rbNQWX

c See https://nyti.ms/3r5ztnf

unreliability—a modern-day Jaguar marketing executive watching the show had "never been happier to see our car not start."[d]

Jobs and Woz at Compaq

Cantwell and Rogers stitched together bits of Don Draper and Steve Jobs to create Joe MacMillan, a brilliant computer marketeer who left IBM under a cloud. It must have seemed like a good fit, given the reputation as an all-time great pitchman Jobs earned while introducing products such as the Macintosh and iPhone. After two movies and a blockbuster biography, Jobs is unquestionably the most famous computing innovator. The early Jobs was a real-life antihero, whose ability to convince others of his own genius created what colleagues called a "reality distortion field." His conviction that he alone knew how things should be done led to disasters as well as triumphs. The fictional MacMillan is likewise prone to inspiring speeches and grand pronouncements but insecure, self-absorbed, and (like Don Draper) haunted by a mysterious past.

Apple Computer's early story provides two archetypal Steves: the slick visionary who promised to "put a dent in the universe" and the hands-on hardware engineer who lived for technical challenges. By the law of narrative templates, where there is a Jobs there must be a Woz. In this case the Wozniak role is filled by unworldly engineer Gordon Clark, suckered by Joe into leading his hardware team. There are not any comparable clichés for female computer engineers, but because an all-male lead cast was out of the question the show's creators adapted the archetype of the manic punk hacker girl, exemplified by Lisbeth Salander from *The Girl with the Dragon Tattoo*, to create the angry and damaged Cameron Howe.

Despite inheriting Jobs' urge to create something insanely great, MacMillan's secret plan turns out to be borrowed not from Apple but from Compaq: create a slightly faster, slightly cheaper IBM PC clone with a built-in screen and carrying handle. The show thus set out to answer a question that nobody has ever asked: What if Steve Jobs and Steve Wozniak started a PC-clone company and hired Lisbeth Salander to write the BIOS firmware code needed to avoid infringing on IBM's copyright? They are clearly the wrong people for that particular job. Creating a successful PC clone meant copying an effective but uninspired design while resisting the urge to make improvements that would compromise compatibility. If Jobs and Woz had founded Compaq rather than Apple then neither they nor it would be remembered today.

Perhaps the showrunners knew this all along and set out to play a long con on viewers who assumed Joe would triumph. It seems more likely, though, that they painted themselves into a corner in the pilot and spent almost an entire season trapped in what the *AV Club* called a "run of alternately humdrum and ludicrous episodes"[e] before realizing they had to blow up their own show to escape. That escape is dramatic but infuriating. Cameron quits after the natural language interface she built for the computer is rejected. The others visit the Comdex trade show to unveil their computer, only to stumble onto a closed-door preview of the Apple Macintosh. Joe, abruptly realizing that his PC clone is not so special after all, then sets fire to the delivery truck holding the first batch in an overly literal interpretation of the show's title. Many reviewers rolled their eyes when Ayn Rand's architect hero destroyed his building because its brilliant design was tampered with. Seventy years later, the narrative gambit had not become any fresher or less juvenile.

The modest satisfactions of the first season come not from the characters but from the computer industry history worked into the background. It takes place not in California but Texas, which in the 1980s was home not just to Compaq but also to Texas Instruments, Tandy, and (a little later) Dell. The characters work at a stable mid-sized company that takes a fateful step into the personal computer market, just as firms like Texas Instruments did in real life. We get to see the cloning of a BIOS, the design of a case, and efforts to procure a display table at Comdex. Even the push for a natural language text interface fits with the mid-1980s effort to build "conversational interfaces" for business systems.

Unexpected Greatness

After its unexpected renewal for a second season "Halt and Catch Fire" improved greatly by pushing the Jobs and Woz archetypes to the sidelines to focus on its female characters. Joe MacMillan is demoted from antihero to manipulative villain. The writers take a less indulgent view of his pathologies as they show him dating an oil heiress to sneak his way back into the computer industry. Gordon Clark spends two seasons as a bumbling supporting character and homemaker.

The show's new central partnership is between Cameron Howe and Donna Clark, wife of Gordon, who spent most of the first season languishing in the Betty Draper template of bitterly neglected spouse with thwarted ambitions. Cameron's brash bleached hair and swaggering aggression are gradually replaced with a much more plausible blend of defensive body language and demure clothing. Together, they found an online games company called Mutiny, modeled on the real-life Commodore 64 service Quantum Link. It begins with the ideal of leaderless hacker collective, which does not prove the most effective management structure.

Like Ullman's memoir, "Halt and Catch Fire" rebuts the technology field's chronic sexism by allowing its female characters to be as interesting, talented, and flawed as any male antihero. That is far more satisfying than simply setting up villainous male foils for them to overcome. The women are not perfect. Cameron, in particular, remains awkward and often selfish even as her character deepens. Her conflicts with the more pragmatic Donna are grounded in a recognizable clash between the narrow idealism of hacker culture and the compromises needed to run a business.

The show continued to improve in its universally acclaimed—yet little watched—third and fourth seasons. Critic Sean O'Neal judged this turnaround an "all-time great creative resurgence."[f] The plot moves fast, apparently because the show runners always expected their small audience to

d See https://bit.ly/3i3jamJ

e See https://bit.ly/3hC84WT

f See https://bit.ly/2ULRFpz

lead to cancellation. I will not spoil its twists and turns with a detailed summary. Halfway through the characters relocate from Dallas to Silicon Valley, though as the entire run was filmed in Georgia there is not a particularly strong feeling of place attached to either setting. The last season jumps forward to the mid 1990s to focus on the early days of the Web.

The four main characters combine and recombine into different combinations of allies and enemies, but all have plausible motives for their mistakes or betrayals and there are no permanent villains. Keeping the same characters through these transitions requires them to demonstrate an implausible range of skills. Cameron, for example, jumps from BIOS coding to natural language processing, before settling down as a creator of video games and online services. Gordon is a microcomputer designer and electronic engineer who can reconfigure an IBM mainframe from batch operation to timesharing during a single evening or hack away to produce a pathbreaking antivirus engine.

As O'Neal noted, by its end "Halt and Catch Fire" had more in common with the funeral home family drama "Six Feet Under" than with "Mad Men." Like "Six Feet Under" it followed "perpetually unsatisfied people" whose hunt for "life-changing leaps" causes them to "shut out and repeatedly hurt the ones they love." Both shows overcame their gimmicky origins "to find their greatest resonance in small, quietly devastating, human interactions."[g] Even Joe MacMillan eventually deepened into a recognizable and sympathetic human being with an awareness of his own flaws.

"Halt and Catch Fire"'s engagement with videogaming as a community-building activity is particularly effective. Toward the end of season three, Cameron explains the premise of her work in progress: a woman travels alone through space on a motorbike in search of five power-ups: a sense of proportion; a sense of humor ("to fend off most forms of attack"); a sense of self ("to appear and disappear"); decency; and "common sense, which lets her see everything more clearly." Coming after a devasting conflict in which Cameron's lack of these qualities hurt her and others, this presentation of game creation as a possible venue for psychological growth seemed like a rejoinder to Sherry Turkle's claim (discussed in my April "Historical Reflections" column) the journey of hackers to psychological maturity had been halted by a wrong turn into subculture that prized technological mastery over human connection.

Like Ullman's memoir, "Halt and Catch Fire" rebuts the technology field's chronic sexism by allowing its female characters to be as interesting, talented, and flawed as any male antihero.

"Halt and Catch Fire" builds a rich bench of supporting characters, including Bos, a folksy Texan financial executive nearing retirement age, and Diane, a successful Silicon Valley venture capitalist. Donna becomes her protégé, giving an almost unique fictional portrayal of venture capitalists as sympathetic characters. Even the soundtrack, consistently well-chosen and usually spot-on for the time periods in question, relaxes. The first season has a punk backing, from the likes of Hüsker Dü and Bad Brains. This mellows noticeably as the show progresses, to the extent of wrapping the final episode with Peter Gabriel's conspicuously unaggressive "Solsbury Hill."

Complicating Success and Failure

Writing this series of *Communications* "Historical Perspectives" columns has made me realize how much we lost when the outsized economic importance of tech from the late 1980s onward has focused most narratives of technological creativity on the pursuit and, usually, accomplishment of enormous wealth and power. In contrast, "Halt and Catch Fire"'s depiction of technological careers driven by personal insecurity, a desire to build something new, and an attraction to constant change resonates with Ullman's memoir, Levy's depiction of hacker culture, and above all with Kidder's depiction of the Eagle team. Winking to this, in one of the show's final shots the camera pans slowly past a first edition of *The Soul of a New Machine*.

When the series finished, James Poniewozik wrote in the *New York Times* that its central concern turned out to be "failure as a condition of human growth."[h] It is worth remembering, though, that these careers only look like failures when set against outsized definitions of success. Between them, "Halt and Catch Fire"'s four core characters found companies based on concepts an awful lot like Apple, Compaq, Dell, McAfee, AOL (as Quantum Link rebranded itself), eBay, PSINet, Netscape, Yahoo, and Google. The implausible breadth of good ideas they failed to fully exploit underlines the fact that for every famous company there were a dozen others with the same idea that are now forgotten. Statistically speaking, you were more likely to be an early employee of Excite, Lycos, or AltaVista than of Google; more likely to found Eagle Computer, Sirius Systems Technologies, or Leading Edge than Compaq. The main characters all achieve some respectable paydays. Some of them live in big houses and drive flashy cars. It is just that their lives look more like Ellen Ullman's than Bill Gates's.

I suspect this dominant narrative of successful computing careers as pathways to world domination may itself have played a part in the field's failure to attract and retain women. I wish we had more great books and shows focused on the processes, challenges, and satisfactions of more attainable technological careers. That might help more people of all genders to imagine themselves working successfully in the field. "Halt and Catch Fire" ends with a moment of potential as Donna shares a new idea with Cameron, but a few minutes earlier it allowed itself a more pointed message. Donna, by now the managing partner at her firm, con-

g See https://bit.ly/3xCUcRJ

h See https://nyti.ms/3xFc5PV

venes a swanky backyard networking event for women working in Silicon Valley. Her reflection on the personal price she has paid for her success in becoming "a partner by trade and a mother and a sister by design" ends with a prediction that her teenage daughters, by then important characters in their own right, will have no need for such gatherings in their own careers. To us, 25 years later, that broken promise lands like a slap to the face.

Further Reading

Ellan Ullman has written no other books comparable to *Close to the Machine* but she did publish a collection of magazine essays written over several decades, many of them on related themes, in *Life In Code: A Personal History of Technology* (Farrar, Straus and Giroux, 2017). According to Ullman *The Bug* (Doubleday, 2003) began as an autobiographical account, becoming a novel when she decided to fictionalize her experience and give the central trauma, a long struggle to locate a simple bug in an early graphical user interface, to a male protagonist. *The Bug* is another notable portrayal of the work of programming, though I personally found it less compelling than her memoir. Perhaps the external viewpoint makes her tortured surrogate, Ethan Levin, more difficult to sympathize with.

Richard Powers has earned a reputation as the contemporary novelist most inclined to take science and technology seriously. His breakthrough book *The Gold Bug Variations* (William Morrow, 1991) focuses on the cracking of the genetic code in the 1950s, but a parallel narrative set in the 1980s includes some great descriptions of IBM mainframes based on his own experience as a programmer and operator. His later *Ploughing the Dark* (Farrar, Straus and Giroux, 2000) centers on a young woman developing a virtual reality system for a Microsoft-like company, but I found it much less convincing. *Gain* (Farrar, Straus and Giroux, 1998) barely mentions computers but comes closer to the spirit of Ullman's book by intertwining the history of a fictional Midwestern chemical conglomerate with the interior perspective of a woman fighting cancer. If you enjoy stories about the intertwining of human flaws and scientific creativity you may also like Allegra Goodman's *The Intuition* (Dial Press, 2006), a novel that tracks the work of a 1980s cancer research team riven by an allegation of misstating the results of a promising treatment.

Douglas Copeland is not a programmer, but he is a snappy describer of pop culture artifacts. Not long after coining the term "Generation X," his quest for the generational zeitgeist took him to Microsoft and Silicon Valley in the mid-1990s for his novel *Microserfs* (HarperCollins, 1995). The opening, describing a communal house full of Microsoft workers, is wonderfully zesty (and can be read on the *Wired* website), though the book starts to bog down once they decamp to California to build something that, in retrospect, seems a lot like Minecraft. C

Thomas Haigh (thomas.haigh@gmail.com) is a professor of history at the University of Wisconsin—Milwaukee and a Comenius visiting professor at Siegen University.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Project-ID 262513311—SFB 1187 Media of Cooperation.

ICCQ

www.iccq.ru
CfP closes on 18 Dec

The Second International Conference on Code Quality (23 Apr, online)

In cooperation with
ACM SIGPLAN and SIGSOFT
IEEE Computer Society

Static/Dynamic Analysis, Program Verification, Bug Detection, and Software Maintenance