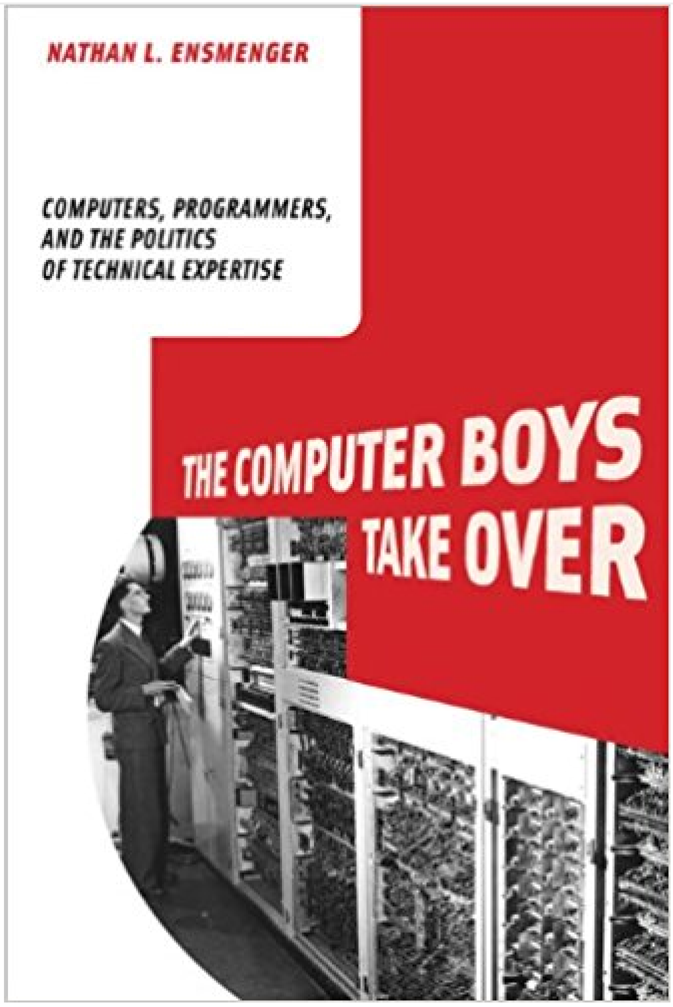
{essays in history}

The Annual Journal produced by the Corcoran Department of History at the University of Virginia

The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise


NATHAN L. ENSMENGER
COMPUTERS, PROGRAMMERS, AND THE POLITICS OF TECHNICAL EXPERTISE
THE COMPUTER BOYS TAKE OVER

Volume 44 (2011)

Reviewed Work(s)

The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise. By Nathan Ensmenger (Cambridge: MIT Press, 2010). Pp. 320. Hardcover, $30.00.

Over the past few years, the study of the history of computing has undergone a rapid expansion. Nathan Ensmenger redirects and challenges this emerging field by calling for a greater methodological emphasis on software history with *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technological Expertise*. In his introduction, he observes two distinct narratives in the history of computing: on the one hand he criticizes hardware-based analyses as presenting, a determinist narrative of constant, unimpeded technological progress. On the other hand, he presents the history of software, one that he argues is “replete with tension, conflict, failure and disillusionment” (10). “In many respects,” according to Ensmenger, “it is the history of computer software and not of the computer itself that is at the heart of the larger story of the great computer revolution of the mid- and late-twentieth century” (5). Ensmenger focuses on the history of computing through the lens of software design, presenting a history of the formative years of computer programming. Between the 1950s and 1970s, corporate managers, programmers, and computer scientists grappled over the nature of computer programming.

During the 1940s, computer programming was initially conceptualized as clerical work. Scientists and engineers dictated what they wanted a given computer to accomplish, and programmers carried out their instructions. As a result, the earliest programmers were women. However, the transition from thought and action through the language of computer code was not as simple as initially envisioned, and it became apparent that programming required specialized skill. Coding on early computers proved a complex, challenging process compounded by fickle hardware and frequent computing errors that involved hours of tedious “debugging.” By the 1950s, “Skilled programmers developed a reputation for creativity and ingenuity, and programming was considered by many to be a uniquely intellectual activity, a black art [as described by FORTRAN developer John Backus] that relied on individual ability and idiosyncratic style" (29). As the prestige of computer programming increased, observers assigned it masculine qualities; gradually, the "computer boys" took the place of the "ENIAC girls."

Between the 1950s and 1970s both corporate managers and the programmers themselves engaged in a complex negotiation over the nature of programming work. For companies such as IBM and RAND's System Development Corporation, there emerged a pervasive sense of a "software crisis," as mangers simultaneously bemoaned the long, expensive process of coding and debugging and the scarce supply of competent programmers relative to the demand. To ameliorate the former, companies adopted programming languages as cost-cutting measures aimed at streamlining the coding process and creating universal bodies of knowledge. For the latter, they vetted candidates through using aptitude and personality tests aimed at filtering people who possessed the predisposed traits of a competent programmer. These measures, Ensmenger argues, were attempts to impose Taylorist and Fordist notions of efficiency to simultaneously "mass produce" computer programmers while imposing managerial control over their labor. Ultimately these measures proved ineffective: languages made coding easier for programmers, but their adoption was usually a matter of personal preference. Meanwhile, the tests proved easily compromised, and their heavy emphasis on mathematics proved impractical for applied programming. These tests, however, did do much to set the terms of what made an "ideal" programmer, establishing the pervasive stereotype of the neurotic, antisocial, male programmer that endures to this day.

As managers attempted to shape programming to fit traditional corporate structures, the programmers themselves attempted to gain control over their own labor. According to Ensmenger, "Programmers were well aware of their tenuous professional position, and they struggled to prove that they possessed a unique set of skills and training that allowed them to lay claim to professional autonomy" (169-170). They attempted to do so by establishing qualification standards, imposing certification programs, and forming professional organizations such as the Association for Computing Machinery (ACM) and the Data

Processing Management Association (DPMA). Their ultimate goal was to distinguish their work as artisanal, as opposed to that of mere technicians. By 1968 the perception of programming as a black art lost its luster, and programmers emphasized their labor as “software engineering.”

However, the field proved fraught with division which ultimately undermined efforts to create universal professionalization standards. Ensmenger pays special attention to the rift that developed between professional programmers in the corporate world and academics in computer science programs (the field of computer science, which emerged from various technical and scientific fields, developed with a heavy emphasis on theory over application with the algorithm, as opposed to the computer, as the fundamental intellectual model). “Computer scientists expressed disdain for professional programmers, and professional programmers responded by accusing computer science of being overly abstract and irrelevant” (129). The theory-intensive ACM and the business-oriented DPMA clashed over how industry standards should take shape, and ultimately these differences could not be resolved. In the end, the aforementioned attempts at professionalization did not pan out, and the very nature of the programmer’s labor proved a complex, contentious battleground.

There is much of value to take from *The Computer Boys Take Over*. In addition to computer historians, historians of technology (especially those interested in labor and skill), and interested popular readers will find that Ensmenger has put together an accessible and fascinating read. His recurring emphasis on the “software crisis” is well-argued, and he demonstrates that it endures to the present. While he cites numerous secondary works, he draws upon a fair amount of primary material. His use of industry magazines and conference proceedings capture the complex tension that characterized the period, and popular magazines from *Fortune* to *Cosmopolitan* address the developing popular perception of computing. Nevertheless, Ensmenger has produced a fresh, valuable addition to a rapidly developing historiography.

Kevin Impellizeri

University of Delaware

Published by

View all posts by

Book Reviews

20th century, Politics, Technology

COMMENTS ARE CLOSED.

FIND US ON:

» Facebook

» Twitter

ADMIN:

» Log in

» WordPress.org