Reviews

Raúl Rojas, Editor
Freie Universtät Berlin

Jon Agar, *The Government Machine: A Revolutionary History of the Computer*, MIT Press, 2003, viii + 554 pp., $52.00, ISBN 0-26-201202-2.

Jon Agar's *The Government Machine* is a path-breaking book that's as much about politics and government as it is about information technology, and it will attract readers from a much wider community than historians of technology and computing. Even for readers who happen to be familiar with both the British political and bureaucratic milieu and with the development of organizational information processing, this book is challenging. The challenge comes from both the sophistication of Agar's historical metaphor and the sheer factual density of this 550-page book.

The book is primarily about the institutionalization of information processing in the UK Civil Service, the executive arm of the British government. We commonly use machine metaphors for such bureaucracies and use terms like "the levers of power" without ever realizing we are speaking metaphorically. Agar's organizing principle in the book is to take the machine metaphor literally. He explains,

> The Civil Service was a general-purpose "machine" governed by a code. The stored-program computer is a general-purpose machine governed by a code. Is the similarity a coincidence, or is there a profound connection? (p. 391)

He then answers his own question: "the computer is indeed a materialization of bureaucratic action, but the connection between the two is far from simple."

The idea that computer architectures mirror the organizations that created them isn't new. It has been said, for example, that the centralized mainframe owed much to IBM's East Coast bureaucratic hierarchy, while the PC reflected Silicon Valley's fragmented entrepreneurial firms. These observations usually come from journalistic accounts of computer history, and no real evidential basis exists for the assertions. By contrast, Agar has taken great pains (in Chapter 1) to discuss the notion of historical metaphor and to validate his approach.

Two interesting outcomes of Agar's approach are alternative explanations for the genesis of Babbage's Analytical Engine and the Universal Turing Machine. Thus, because Babbage described his Analytical Engine as separating a computation's "legislative" (coding) and "executive" (processing) parts, "if we take Babbage at his own word, the Analytical Engine must be seen as a political machine" (p. 41). This is not the place to take issue with Agar on details, but suffice it to say that Babbage scholars have more plausible explanations for the Analytical Engine's origins. In the case of Turing, Agar argues that his family was steeped in the culture of the Indian Civil Service and that Turing "drew on features of British bureaucracy as a resource to articulate the range of operation of the universal computing machine" (p. 268). No doubt Turing scholars will smile on Agar's construction and move on.

In fact, the discussions of Babbage and Turing fit somewhat uncomfortably in the book, which is largely about the development of government information processing systems during the 19th and 20th centuries. These were undertakings in which millions or billions of data items were processed, and they were very different in scale and spirit to Babbage's and Turing's inventions.

The modern Civil Service took shape in the mid-1800s. Activities such as census data processing were carried out by large numbers of unskilled clerks organized by an elite managerial class (hardware and software, in Agar's metaphor). Although entirely unmechanized, the information processing systems—the software—were highly sophisticated. A major theme of the book is the part that "experts" played in shaping and mechanizing these information systems. Agar shows that British experts played the same role as "systematizers" in the transformation of the American office in the 1890s, which has been well described in the writings of JoAnne Yates. In the 1930s, these experts became known as the Organization and Methods (O&M) Group, and in 1979, this became the independent Central Computer and Telecommunications Agency. Since the 1990s, most government IT projects have been outsourced to the likes of EDS and Accenture. Agar shows that the role of the expert in these evolving organizations has been persistent.

The book is largely a set of case studies, chronologically arranged, and linked by the themes of the mechanical metaphor and expert movements. Several of the examples—such as the Central Statistical Office and Bletchley Park—are well known and researched, but Agar brings new insights. For example, when describing Bletchley Park, he focuses on the information system rather than on the machines—how data flowed in the organization, how it was captured by punched cards and indexes, and how salient facts were extracted. This is a topic that is almost completely ignored in the more tedious accounts of cryptanalytic machinery, and it shows the power of Agar's metaphor: it forces him to address issues that others have not even perceived.

In other cases, Agar has broken new ground, using material from the British National Archives. Among the

1058-6180/06/$20.00 © 2006 IEEE

more interesting case studies is that of British air defense. In World War II, "spotters" telephoned the details of incoming aircraft to a command and control center, where aircraft were represented by colored counters on tabletop maps of the airspace. Between observation and display, however, there were filtering and validation operations that reduced the noise in the system. The system was largely manual but elaborate. In a subsequent chapter, Agar carries the story forward to Britain's postwar computerized defense systems (based on secondary sources, as primary documents have not yet been released).

Another set of case studies addresses national registration, and Britain's 100-year opposition to identity cards. This is a complex story. Because national registration was politically impossible, except in time of war, the authorities turned a blind eye on "data creep" (using personal data for purposes other than what it was gathered for) and managed by integrating data from existing registries and databases (such as national health records, police records, and driving license records). In this subtle and political story, Agar captures Britain's knack for muddling through. The whole book is a very British story.

In this review, I have touched on only a few of the many themes, cases studies, and subplots in this book, and if the review lacks coherence, so too does the book. The book contains many themes fighting to get out, many cross-references to previous chapters, and the organization is so loose that it's a truly challenging read. It is a book of genuine ambition, but coherent it is not. My advice is to read it, nonetheless. Agar's approach might change the way you think about the history of computing. In its holistic, and occasionally contentious, view of computing, it's almost in a class by itself (only Paul Edwards' *Closed World* comes close). It's a wonderful example of how our field of study is maturing. There will be better books, for sure, but *The Government Machine* will be one of the books that touched the tiller on the course of computer history.

Martin Campbell-Kelly
University of Warwick
http://www.dcs.warwick.ac.uk/~mck/Contact.html

Andy Hertzfeld, *Revolution in the Valley: The Insanely Great Story of How the Mac was Made*, O'Reilly, 2005, 290 pp., US$24.95, ISBN 0-596-00719-1.

It's been said that the Apple Macintosh is not a computer but a religion. If so, this book is the equivalent of the Bible. *Revolution in the Valley* is not a scholarly work of historical investigation; it reads more like transcribed oral history—that is, a collection of fragmentary accounts written by Andy Hertzfeld and some other members of the original Macintosh team. The Macintosh community, just as any other cultural community, can now look back to its "founding myth"—a saga of creation, full of heroes, villains, and epic struggles. Some of the main characters in the book are nowadays millionaires and have retired from the computing business. Now they have the time to look back at the work they did in their 20s and marvel at what they accomplished—namely, creating a line of computers that still exists and is the only surviving alternative to the all-encompassing Windows-Intel monopoly. Gone are the Apollo workstations, the Osborne computers, the DEC machines, the Commodores, even the Compaqs, but the Mac is still here.

The original Macintosh is by now legendary. All those already into computing in the 1980s will remember the launch of the machine: a 30-second commercial, aired during the 1984 Super Bowl, compared IBM to Big Brother, and closed assuring us that the year would not be like Orwell's *1984* because of the Macintosh. A few days later the machine was publicly shown; the Mac was what Microsoft Windows would not become until 1995. The Mac had an intuitive and pleasing GUI, a mouse, and applications such as MacPaint and MacWrite that made the computer useful at home and in the office. How Apple could squander a 10-year advantage over Microsoft is one of the mysteries and great tragedies of the computer industry. However, this book is not about that failure, it's about the joy and passion of creating the machine.

The most astonishing single aspect of this volume is the youth and inexperience of the original Macintosh team. The Lisa (in some way the direct predecessor of the Macintosh) was designed and built by more seasoned professionals, some of them hired directly from Xerox, where the mouse and GUI had been invented. Yet, the Mac team was a crew of juveniles playing video games at night. As Hertzfeld notes, the group was more like a start-up company inside Apple. The Mac team comprised a bunch of irreverent programmers, a group of idealists who coalesced around Steve Jobs with the mission to design an "insanely great" machine.

The original idea for the Macintosh came from Jef Raskin, who wanted to build an affordable and easy to use computer—a computing appliance, as we would say today. He recruited some of the early Mac designers, especially Burrel Smith and Bud Tribble. However, Raskin left Apple in 1981, and Steve Jobs immediately

adopted the project. From the stories collected by Hertzfeld, it's obvious that there was a management hierarchy at Apple that Jobs and the Macintosh designers successfully bypassed. When the team wanted something, they would usually talk directly to Jobs, who would grant or deny their wish. Everybody seemed to be working after hours in the project, and Jobs would usually visit in the afternoons to assess the team's progress. Although revered by the Macintosh developers, Jobs emerges in this book as the egomaniac he reportedly is, always protected by his own "reality warping field."

Three people stand out in this book: Hertzfeld, because he wrote important parts of the operating system's core; Burrell Smith, who did the main design of the Macintosh logic board; and Bill Atkinson, who programmed the graphical toolbox for the Lisa computer and then for the Mac. Many others are mentioned in the book, who also had an important contribution, but these three even appear sitting in the middle of the picture on the book's cover.

Burrel Smith was hired by Apple as a service technician. He had no formal academic training and became a self-taught computer designer. As he would write later, in a form he had to fill about his career, he was trained at the "Steve Wozniak University"—that is, in the larger university of life. Herzfeld relates how Smith wired the Mac prototype board by hand and how Bud Tribble convinced him to drop the 8-Bit 6809 microprocessor and adopt instead the newer 68000 processor from Motorola. The decision made the machine more expensive but also more powerful and even faster than the Lisa, which was clocked at 5 MHz. (The Mac was clocked at 8 MHz.) Using programmable logic arrays (PALs), Smith was able to iterate the different board designs faster than would have been possible using logic functions on chips. When finished, the board was a work of art, one on which even Wozniak would have taken pride.

The stories in the book (which Hertzfeld originally collected on a Web site) make clear why the team made some decisions, both right and wrong. The screen of the Macintosh had square pixels, it could be repainted fast, and its fonts were agreeable to the eye, but it was too small for serious business applications. A choice that almost killed the Macintosh was not to provide any slots for third-party hardware. This decision was made mainly by Jobs, who did not want the user to fiddle with the machine's innards. He wanted the programmers to always have a well-defined environment they could trust and program for, eliminating the complexities of interfacing hardware to software drivers. Jobs even prohibited the hardware designers to include additional addressing lines for a memory upgrade from 128 to 512 Kbytes. If users wanted the additional memory, they would have to buy the 512-Kbyte model. The designers left the additional lines without telling Jobs, and the first generation of savvy Mac users found out fast that the more capacious memory chips could be soldered piggyback on top of the less capacious memory chips (as many of us really did!).

Atkinson is maybe the most colorful character in the book. He was an artist–programmer who single-handedly wrote the graphic library and the MacPaint drawing program. One story tells how offended he was when the Lisa was unveiled, and his name was not mentioned to the press, although he had written most of the graphical interface. Atkinson took Hertzfeld to see Jobs and complained very emotionally. Although Jobs dismissed him, he rapidly arranged for Atkinson to be made an Apple Fellow and let him put his name and icon in a MacPaint menu. Some years later, Atkinson wrote the HyperCard application, where hyperlinks let a user navigate in an information space. It was like HTML before the Web.

Bill Gates does not fare well in the book. Hertzfeld describes his tough tactics and how, when confronted by Jobs about being in the process of stealing Apple technology (for what would later become Windows), he answered that, yes, he had broken in to steal, but Apple was already holding the bounty stolen from Xerox. Unfortunately for Apple, the company gave Microsoft a perpetual license to the Mac user interface at the time when Apple depended on Microsoft software for the Apple II, the bread and butter of the company before Macintosh sales took off. Microsoft was able to steal the crown jewels from Apple before anyone inside the company could realize what would happen further down the road.

Although some of the stories in the book have already appeared in other collections and books about Apple, this is the most compact (I read the book in one evening) and, I would say, most reliable set of Mac folklore tales. The book is lavishly illustrated with photographs, scanned pages of handwritten code, to-do lists, and photographs of the evolution of the Mac user interface. Those who can recognize programming elegance by looking at fragments of code will be amply rewarded with the many facsimiles of Andy Hertzfeld's coding notebooks.

Revolution in the Valley is a must for any Macintosh owner, coming from the pen of the original Mac wizards. It's also a source of first-

hand material about the conception and birth of the Macintosh, one of the great computers of the 20th century and, who knows, maybe also the 21st century.

Rául Rojas
Freie Universtät Berlin
rojas@inf.fu-berlin.de

Raul Rojas, Director: 1+1: Tres.Â Looking for some great streaming picks? Check out some of the IMDb editors' favorites movies and shows to round out your Watchlist. Editors' Picks: Old School Cool. Iconic Summer Movie Picks. Pride: LGBTQ+ Stories in Film and TV. RaÃºl Rojas is Professor of Computer Science at the Free University of Berlin. Ulf Hashagen is affiliated with the Munich Center for the History and Science and Technology, Deutsches Museum. Read more. Product details. Series: History of Computing. Hardcover: 457 pages. Publisher: The MIT Press; 1st edition (July 7, 2000).