Knowing with Computers: How Software and Systems Encapsulate Expertise

These papers use the lens of practice to examine one of the key concepts in computer science: the abstraction and encapsulation of complexity. Any modern computer system consists of dozens of layers of abstraction, with each level of software hiding its actual workings from those above it. This has profound implications for the ways of knowing available to the users of computer systems. Even programmers and system designers interact not with the computer itself but with virtual machines and application programming interfaces. Social assumptions, specialist expertise, and epistemological constructs are embedded within these virtual machines. Haigh and November both examine attempts to embed specialist mathematical knowledge in reusable software routines, intended to make mathematical techniques more accessible to users without the knowledge to understand their inner workings. McMillan explores efforts during the 1960s to create programming languages better able to support this kind of abstraction and code reuse, tying this approach to the rise of computer science with its focus on mathematical logic. Jesiek examines the origins of "embedded systems" (as the specialized computers built into cars, DVD players, etc are known), looking at their blurring of hardware and software design, hiding from the user not only knowledge of the computer's workings but even awareness that they are dealing with a computer. Kita considers the power of user interfaces to encapsulate revolutionary technology within a familiar package, using an early Japanese airline reservation system as an example.

Thomas Haigh, thaigh@computer.org
University of Wisconsin, Milwaukee
"Knowing Numbers: How Numerical Software Libraries Changed Scientific Practice, 1954-1975"

This paper explores the social and epistemological consequences of the adoption of mathematical software packages by scientists and engineers from the 1950s to the 1970s. Certain mathematical operations, most importantly the solution of differential equations and the manipulation of large matrices, appear in many different disciplinary fields. Using electronic computers it was possible to produce numerical approximations thousands of times faster than before, but this increase in speed exposed the limitations of traditional mathematical methods in which scientists and engineers were trained. New, more accurate methods were devised but these were hard to implement, requiring expert knowledge of both computer architecture and specialized areas of applied mathematics. During the 1960s, major laboratories such as Los Alamos, NASA's Jet Propulsion Laboratory, and Argonne had created standard libraries of mathematical routines to implement common functions. By the 1970s several businesses were selling these libraries commercially, moving certain kinds of mathematical expertise out of the laboratory entirely. I examine the consequences of this shift for scientific practice, focusing on the black-boxing of mathematical expertise into software, the creation of a new community of mathematical software specialists, and the consequences for scientific education and the social organization of major laboratories. Sources include oral history interviews, archival records, and conference proceedings.

Joseph November, november@gwm.sc.edu
University of South Carolina
"Computers and the Unintended Dematematization of Biology"

In 1960, when the directors of the National Institutes of Health (NIH) began to commit major resources to the introduction of computers to biology and medicine, they regarded the computer as the means to mathematize the life sciences. To the NIH's vast disappointment, it was clear by the late 1960s that the presence of computers was having the opposite effect in laboratories and hospitals: rather than enabling biomedical researchers to become more rigorously mathematical, computers were allowing them to black-box many of the mathematical components of their work. To illustrate how this ironic development came to pass, I will examine two areas: 1) the goals and constraints of the NIH's initial vision for biomedical computing; 2) how and why researchers used NIH-sponsored computers at UCLA to create software tools that would obviate the need for life scientists to think mathematically.

Bill McMillan, wmcmillan@emich.edu
Eastern Michigan University
"The Origins of Structured Programming in the Mathematical Abstractions Implemented in the Transition from ALGOL 58 to ALGOL 60"

By the early 1970s, the structured programming movement was reshaping academic computer science and professional practice. This rational approach to software design, which advocated a layered, abstraction-based view of software, was given great impetus and credibility by the NATO meetings on software engineering in 1967 and 1968, and by the recognition that undisciplined use of the GOTO statement had led to much tangled code that was difficult to maintain. The transition to structured programming was anticipated, enabled, and, to an extent, foreordained by the creation in the 1950s of programming languages derived from formal approaches to reasoning in mathematics and logic. ALGOL 58 was inspired largely by FORTRAN and, like FORTRAN, lacked several features critical for the clear expression of complex algorithms. By introducing block structures; local, dynamically-allocated variables; recursion; flexible means of passing data between routines (call-by-value and call-by-reference); and restricted use of the GOTO, ALGOL 60 provided the model of a modern programming language ready to meet the needs of software design widely recognized by the late 1960s. The thinking behind these advances and related ones, e.g. the development of the LISP programming language, followed directly from abstract models of computation such as Church's Lambda Calculus.

Brent Jesiek, bjesiek@vt.edu
Virginia Tech
Embedded Boundaries, Embedded Systems: Historical Trajectories and Contemporary Trends

From the Apollo and Minuteman guidance computers of the 1960s to the specialized electronic chips and circuitry that now reside in our cell phones, cars, and coffee pots, "embedded" or special-purpose computer systems have both a long history and a high degree of contemporary relevance. Yet to date, historians, social scientists, and other scholars have largely overlooked this important domain of technology. In this paper I begin to open up this area of research, first by presenting a brief historical introduction to embedded systems. I then turn more specifically to the emergence and evolution of embedded system design tools and techniques, beginning in the 1970s with the advent of new integrated circuit technologies and hardware design languages (HDLs), and culminating more recently with the software/hardware co-design movement. My analysis pays close attention to the historical negotiation of the fuzzy sociotechnical boundaries around the software and hardware, and users and designers, of embedded systems.
I also document how embedded systems technology and knowledge have been uniquely "black-boxed," especially in comparison with general-purpose computers. I conclude by discussing some of the political implications of my analysis, especially in light of ongoing debates over the merits of "open" verses "closed" approaches to the design and development of embedded devices and systems.

Chigusa Kita, chigusa.kita@nifty.ne.jp
Kansai University, Japan

Familiar Look, Revolutionary Technology

This paper explores the importance of user interface design to retain an interface similar to that used by entrenched systems as an important means of encapsulating new technologies within a familiar and accessible guise. As early as in the late 1950s, researchers at Japan Railroad Company started investigating the possibilities of building an automatic seat reservation system. The leader of the research group was Mamoru Hosaka, who formerly designed airplanes during the World War II. Being an outsider to the computer field, he did not stick to "digitalization" but could establish a unique approach to keep the analogue process to retain the same look to the workers at stations. Because he found the key to the success of the introduction of the new system would depend on the approval by the workers and users, and it is better to keep the proce s as long as possible So he encapsulated new technology in the system when it is possible. For example, this revolutionary system used stamps to print the tickets, instead of inventing a digital printout system which was one of the most difficult problems in Japanese information processing. The new reservation system was a great success, but contemporary researchers in the computer field in Japan could not admit it a "computerized system," but a "supecialized mechanical system."