

Computing
Research Association

Using History
To Teach
Computer Science
and Related Disciplines

Edited by

Atsushi Akera
Rensselaer Polytechnic Institute

William Aspray
Indiana University—Bloomington

The workshops and this report were made possible by the generous support of the Computer and Information Science and Engineering Directorate of the National Science Foundation (Award DUE-0111938, Principal Investigator William Aspray). Requests for copies can be made by e-mailing info@cra.org.

Copyright 2004 by the Computing Research Association. Permission is granted to reproduce the contents, provided that such reproduction is not for profit and credit is given to the source.

Table of Contents

I. Introduction	1
1. Using History to Teach Computer Science and Related Disciplines	1
William Aspray and Atsushi Akera	
2. The History of Computing: An Introduction for the Computer Scientist	5
Thomas Haigh	
II. Curricular Issues and Strategies	27
3. The Challenge of Introducing History into a Computer Science Curriculum	27
Paul E. Ceruzzi	
4. History in the Computer Science Curriculum	33
J.A.N. Lee	
5. Using History in a Social Informatics Curriculum	39
William Aspray	
6. Introducing Humanistic Content to Information Technology Students	61
Atsushi Akera and Kim Fortun	
7. The Synergy between Mathematical History and Education	85
Thomas Drucker	
8. Computing for the Humanities and Social Sciences	89
Nathan L. Ensmenger	
III. Specific Courses and Syllabi	95
<i>Course Descriptions & Syllabi</i>	
9. History in the First Course in Computer Science	95
J.A.N. Lee	
10. History of Computing for Computer Science and Information Systems Students	109
Thomas J. Bergin and Jack Hyman	
11. Enhancing Learning Using the History of Computing: Opportunities to Use History in an Introductory Computers and Information Course	139
Thomas J. Bergin	
12. An Introductory Informatics Course	147
Mehmet M. Dalkilic	

13. Engineering and Computer Science in Action: A Course on “The Structure of Engineering Revolutions” 157
Eden Miller Medina and David A. Mindell

Additional Course Syllabi

14. STS.035: From Analog to Agents: Introduction to the History of Computing 167
(course syllabus)
David A. Mindell
15. Informatics 303: Organizational Informatics 175
William Aspray
16. Informatics 400/590: Internet and Society 179
William Aspray
17. Informatics 400/590: Privacy, Security, and Information Warfare 185
William Aspray
18. HSSC120: Computer Learning for the Arts and Sciences 193
Nathan L. Ensmenger
19. ITEC-1210: IT Revolution—Myth or Reality? 197
Kim Fortun
20. ITEC-1220: The Politics and Economics of IT 217
Atsushi Akera

IV. Historical Case Studies for Computing and Information

Systems Education 225

21. What Was the Question? The Origins of the Theory of Computation 225
Michael S. Mahoney
22. History in the Study of Computer Architecture 233
John Impagliazzo
23. Bringing Women into the Curriculum 239
Jennifer S. Light
24. Human Computers and their Electronic Counterparts 245
David Alan Grier
25. Integrating Ethics into a Computing Curriculum: A Case Study of the Therac-25.... 255
Chuck Huff and Richard Brown

V. Resources for Instructors	279
26. Key Resources in the History of Computing	279
Thomas Haigh	
27. Resources for a Course on the History of Computing	295
Thomas J. Bergin and Jack Hyman	

Appendix

1. List of Additional Workshop Participants	309
---	------------

I. Introduction

1. Using History to Teach Computer Science and Related Disciplines

William Aspray
Indiana University at Bloomington

Atsushi Akera
Rensselaer Polytechnic Institute

This volume on using history to improve computing education came out of conversations that the first author had in 2000 with Andrew Bernat, the current executive director of the Computing Research Association who was then a program officer at the National Science Foundation, on leave from the University of Texas at El Paso. Andy and I had wide-ranging talks about various means for improving computer science education. He encouraged me to submit a proposal to NSF on using history to improve undergraduate computing education and talked with me on several occasions as I was preparing the proposal. Since Andy and I previously had written a report together, he was not permitted, under conflict-of-interest rules, to be the program officer for this proposal. The proposal was turned over to Harriet Taylor, who helped me through the proposal process. NSF awarded grant number DUE-0111938 to support this activity.

The grant was used to hold two workshops: the first at Amherst College August 6-7, 2001, and the second at the University of Minnesota April 26-28, 2002.¹ Each was attended by about forty people, primarily undergraduate teachers of computer science.

The original plan for disseminating the results was to make a few presentations at meetings such as the annual ACM SIGCSE conference or the annual Frontiers in Education conference. The organizing committee's frugality meant that attendees of the first workshop suffered through 100-degree weather in dorm rooms with no air conditioning, and attendees of the second meeting had to trek almost a mile through snowstorms to the meeting space. However, this frugality also meant that we had funds left over to collect and edit papers, publish them, and post them on a website where they could receive much wider dissemination than any workshop talk would achieve.

In meetings held in Chicago in April 2004 and in Needham, Mass. in May 2004, the contributors to this volume met to critique one another's papers. The contributors include most of those who spoke at one or both of the workshops, plus a few additional historians and computer scientists who were invited to submit papers. Professor Atsushi Akera of Rensselaer Polytechnic Institute, who was one of the workshop speakers, kindly offered to take principal responsibility for editing the publication. Jean Smith of the CRA staff has undertaken her usual exemplary work as the copyeditor for the volume, under a very tight set of deadlines.

¹ The programs for those workshops can be found online at:
<http://www.cra.org/Activities/workshops/history/august01.html> and
<http://www.cra.org/Activities/workshops/history/april02.html>

The volume itself is organized into five parts. Part I includes this introductory paper, followed by one by Professor Thomas Haigh that provides a basic introduction to the history of computing, as relevant to computer scientists and professionals.

Part II assembles a set of papers addressing broader issues of curriculum development and strategy. Paul Ceruzzi opens this section with a reflective essay on the challenges of introducing history into a computer science curriculum. This is followed by J.A.N. Lee's broad-based discussion about doing just this, especially in the context of the ACM/IEEE Computer Society's jointly produced report "Computing Curriculum 2001," which specifically calls for broadening a computer science student's education.² William Aspray describes his work in developing a broad, social informatics curriculum at the new School of Informatics at Indiana University in Bloomington. Atsushi Akera and Kim Fortun offer a complementary account of their efforts to develop a coherent curriculum in the humanities and social sciences for information technology (IT) students at Rensselaer.

As well, Thomas Drucker provides an account of the classic synergy between the history of mathematics and mathematics education. In a way, history has been integrated into mathematics far more effectively than into computer science education, so his paper provides a good role model for more intensive interaction between history and computing-related disciplines. Lastly, Nathan Ensmenger describes a special program established at the University of Pennsylvania to provide basic computer literacy to non-computing majors within the School of Arts and Sciences. The premise of this work was that students in fields such as history, anthropology, and comparative literature were even more likely to benefit from using historical sources to develop a different kind of awareness about computing and information technologies.

The papers in this section provide a surprisingly coherent picture of how historical materials can be incorporated into a computing curriculum. Moreover, in describing the curricular needs of computer science departments, information technology programs, and schools of social informatics as well as general arts and sciences, the papers collectively provide an account of the different approaches that need to be taken in supporting the different curricular needs of these programs.

Part III of this volume provides a specific set of course descriptions and course syllabi. Rather than speaking abstractly about how to use history, we felt that a set of fully developed course descriptions and syllabi would be most useful for computer science and other faculty who are serious about introducing historical materials into their courses. We have been careful to select a set of syllabi that reflect different degrees of engagement with history and other humanistic disciplines. To enhance student learning in computer science, we find that it is as important to use historical case studies as it is to use a much more integrated approach that seeks to use humanistic content to help transform the professional identity of various computer-related disciplines.

The primary course descriptions provided in Part III include J.A.N. Lee's efforts to integrate historical materials into an introductory computer science course, specifically in an effort to conform to the ACM/IEEE Computer Society's "Computing Curriculum 2001." Also included is a description of the work of Thomas Bergin and Jack Hyman at American University in developing a history of computing course for computer science and information systems students. This is followed by Thomas Bergin's description of a separate course, where he uses

² ACM/IEEE Computer Society, "Computing Curriculum 2001" (Steelman Report), <http://www.computer.org/education/cc2001/steelman/cc2001/index.htm>.

historical case studies to improve student learning and retention within an introductory computing and information systems course for non-majors. Mehmet Dalkilic—whose students, we have heard, truly admire as a superb instructor—describes a similar effort to use historical materials in an introductory informatics course at Indiana University in Bloomington. Eden Miller and David Mindell then describe an upper division course in which students are taught not to read history, but to write it. The course was developed as part of an institutional push at MIT to broaden a computer science and engineering student's education, especially during the last year of a five-year program for a Masters in Engineering. As suggested by Miller and Mindell, such a course has been very effective in getting students to realize that what they label as just being “politics” is, in fact, integral to their disciplinary practice as engineers.

In compiling this volume, we also decided that it would be useful to have all of the course syllabi assembled in one place, rather than having them appear as separate appendices to other chapters within the volume. In addition to the syllabi for the courses described in Part II by Aspray, Ensmenger, Fortun, and Aker, we have also included in Part III an innovative syllabus for the history of computing at MIT assembled by Mindell.

Part IV of this volume offers instructors an array of historical case studies that could be incorporated into a computer science and related curriculum. Most of the case studies should work in different kinds of courses and curricula. Also, these case studies are primarily meant to illustrate the kinds of case studies that are available. Clearly, we are unable to provide a comprehensive set of case studies in this volume.

In the first of the case studies, Michael Mahoney of Princeton University offers material from his extensive investigations into the history of computer science that might be suitable for incorporation into a computer science curriculum. John Impagliazzo provides similar material from the history of computer architectures, which he suggests can enhance student engagement and retention in a contemporary computer architectures course. The paper is based on his experiences at Hofstra University. Jennifer Light, Northwestern University, draws on material about early women programmers to suggest how understanding the social dynamics of technical labor can provide a deeper understanding of tacit knowledge and the process of technical innovation. She argues specifically for bringing women back into the computing curriculum. David Alan Grier, George Washington University, offers a highly complementary account of human computers in general and their contribution to modern programming practice. His paper is based on a forthcoming book on the general subject of the time when computers were human. Lastly, Chuck Huff and Richard Brown from St. Olaf College revisit a classic case study in engineering ethics, the Therac-25, a medical diagnostic instrument whose software engineering flaws generated intermittent failures that produced numerous fatalities. Through their careful consideration of this case study, Huff and Brown are able to draw out the real nature of the ethical dilemmas integral to software engineering design. They also suggest ways in which a “socio-technical” view of engineering practice can be integrated into other computing-related curricula.

The final section of this volume, Part V, provides a set of complementary resources for instructors. Thomas Haigh's “Key Resources in the History of Computing” offers an extensive list of current books, articles, websites, and other material for instructors who wish to find relevant historical material for their computing and computer science courses. Haigh has kindly annotated this selected compilation of sources, and has done so in a lively style. All instructors, including historians teaching a course in the history of computing, should find this to be a valuable resource. Also, Thomas Bergin and Jack Hyman have developed a truly extensive online resource for teaching the history of computing. In addition to a variety of visual materials,

they have placed online all of the lectures and handouts that they have used in their history of computing course. Whether or not these lectures are used in their entirety, the array of images contained in the lecture slides should be an indispensable resource for instructors seeking to locate such images.

In addition to NSF and all of the people named above, we would like to thank Dana Neill, the business manager at the Computing Research Association, for meeting arrangements and other administrative support; Burt Grad, head of the Software History Center, for allowing us to piggyback our May 2004 meeting on his software history conference as a means to attract more historians to help with review of the draft papers; and the staff of the Charles Babbage Institute, in particular Beth Kaplan, the archivist, for hosting the meeting in April 2002.

2. The History of Computing: An Introduction for the Computer Scientist

Thomas Haigh
University of Wisconsin-Milwaukee

Abstract

This paper is intended to provide an introduction to the study of the history of computing for someone who has a good knowledge of computer science, but only a limited knowledge of the history of computing and no formal training in any kind of historical study. It has three parts. The first outlines the development and current state of the history of computing as an area of academic study. The second examines history as an academic field of study, outlining its research methodologies, the various historical subfields relevant to computing, and the ways in which historical research differs from computer science research in its goals, assumptions, and practicalities. This section also specifies in general terms which areas of the history of computing have been studied so far and which ones are currently languishing in obscurity. The final section offers some personal thoughts on different ways in which attention to the history of computing might help to make a computer scientist into a better teacher and, on a broader level, might make computer science itself into a better discipline.

This paper should be read in conjunction with my detailed description of “Key Resources in the History of Computing” (see chapter 26). Where that guide provides an annotated list of specific online, published, and institutional resources in the field, this paper is more concerned with giving the reader a sense of what it is that historians of computing do, why they do it, and how they are different from computer scientists.

The paper has three parts. The first outlines the development and current state of the history of computing as an area of academic study. The second examines history as an academic field of study, outlining its research methodologies, the various historical subfields relevant to computing, and the ways in which historical research differs from computer science research in its goals, assumptions, and practicalities. This section also specifies in general terms which areas of the history of computing have been studied so far and which are currently languishing in obscurity. The final section offers some personal thoughts on different ways in which attention to the history of computing might help to make a computer scientist into a better teacher and, on a broader level, might make computer science itself into a better discipline.

The History of Computing

Earliest Developments

The study of the history of computing is at least as old as the electronic computer. The creation of the first experimental programmable electronic computers in the late 1940s, followed by the first commercially produced models in the early 1950s, created a market for books and articles explaining these electronic marvels to a bemused public. The authors of these pieces wasted few superlatives in celebrating the unprecedented speed and power of these machines. Indeed, the earliest and most influential of the books was entitled *Giant Brains, or Machines That Think*, a title that more sober computer experts spent decades trying to dispel from the public imagination.

Wide-eyed awe, however, only got the popularizers so far. In trying to explain what the new computers did and how they did it, the early computer promoters and salesmen quickly turned to homely analogies and comparisons with earlier and more familiar technologies. Because the first computers were designed and used to perform scientific calculations, Edmund

C. Berkeley, author of *Giant Brains*, presented the computer as the latest in a long and unbroken line of calculating aids that reached back through the slide rule and desk calculator to the abacus. This invocation of Pascal's calculator and Napier's bones became a familiar part of introductory guides to computing, presented in television series, children's books, and public exhibitions. With the success of electronic computers came minor celebrity for nineteenth century British inventor Charles Babbage, whose unsuccessful attempts to construct a programmable computer had been forgotten long before his death in 1871.³

Until the 1970s, the history of computing was largely confined to this role as the source of material for the introductory pages of an article or chapters of a book, designed to make the unfamiliar less threatening. But as the pioneering generation of academic and commercial computer experts began to sense their own mortality, some of them developed a new concern with documenting and preserving the early history of electronic computing, such as the ENIAC project, the first commercial systems produced by IBM and UNIVAC, and the work of computing teams at places like the RAND Corporation and the Institute for Advanced Study.

The first major organized project in the history of computing was an extensive series of oral history interviews in the early 1970s. The project was sponsored by AFIPS, the now-defunct consortium of computer-related academic societies, and was carried out in collaboration with the Smithsonian Institution. These conversations with many of the most important figures in computing from the 1950s were recorded and transcribed for posterity. Following the pattern of most early work in the area, the project leaders were concerned primarily with hardware, and paid little attention to software and still less to the usage of computer systems.⁴ Around the same time, some participants began to turn their attention to memoir-writing, and in some cases became keen amateur historians of their own field. Among the most prominent examples was Herman Goldstine, a collaborator of John von Neumann in the very early days of electronic computing, who wrote an important book called *The Computer from Pascal to Von Neumann*, which blended the earlier history of computing with his own perspectives on events of the 1940s.

Three Key Institutions for Computing History

The three most important institutions in the history of computing all date from the late 1970s. The first of these is the journal *Annals of the History of Computing*⁵ (now *IEEE Annals of the History of Computing*). Originally published by AFIPS, *Annals* (as it is informally known) has developed along with the community it serves. Its first editor-in-chief, Bernie Galler, and almost all the active members of its original editorial board were distinguished computer scientists. These included software engineer Brian Randell and COBOL designer Jean Sammett. The journal printed a wide range of material, running memoirs and anecdotes alongside scholarly articles, printing the transcripts of reunions, devoting special issues to historically significant machines such as the IBM 650, and even featuring a regular section entitled "Meetings in Retrospect." Its rare controversies were usually personal in nature, the most heated involving questions related to the invention of the computer.

³ Berkeley, Edmund C., *Giant Brains or Machines That Think* (New York, 1949).

⁴ These oral history collections are still available for reference in the Archives Center of the National Museum of American History <http://americanhistory.si.edu/archives/ac-i.htm>, though at the time of writing almost none exist in electronic form.

⁵ <http://www.computer.org/annals>.

While *Annals* today still contains many of these elements, it has gradually evolved into something more akin to a conventional academic journal. The demographic balance of its editorial board has gradually shifted away from eminent computer scientists and toward a younger group trained as historians. Memoirs, while still common, are increasingly being joined by more scholarly and analytical peer-reviewed articles. There has been a corresponding, and far from complete, shift away from articles concerned primarily with documenting stories that would otherwise be lost and toward those attempting to place events in a somewhat broader social or intellectual context.

The second major institution in the history of computing is the Charles Babbage Institute.⁶ The Babbage Institute found a permanent home at the University of Minnesota campus in Minneapolis, thanks to the support of the university and of the once-thriving local computer industry. Though it has remained quite small (with a full-time staff of about half a dozen), CBI provides vital services to the historical community. Its primary role is as an archival repository. On a physical level, archives typically consist of a large, climate-controlled storage area in which boxes full of papers, photographs, and other historical materials are housed. Researchers request particular boxes and consult them in a small reading area. The archive is responsible for acquiring, sorting, and cataloging the papers, which are generally donated from the internal files of firms and other organizations or the personal collections of individuals. Historians generally use archives as the main source of material for their research. A steady trickle of acquisitions has given CBI an unrivalled collection of computer-related materials, from large collections such as the records of computer manufacturers Burroughs and CDC and of the Data Processing Management Association, to smaller collections such as the personal correspondence of Daniel McCracken, an early computer textbook author and ACM President.

CBI also houses the largest collection of computer-related oral history interviews, almost all of which have been transcribed and abstracted. The full text of most of these interviews is available online, including such luminaries as Donald Knuth, Marvin Minsky, Gene Amdahl, Thomas J. Watson, Jr., J. Presper Eckert, and Edsger W. Dijkstra. Over the years, CBI and its staff have received a number of research grants, including one to examine the role of ARPA in fostering the development of computer science, and another to explore the history of software. A reprint series, now dormant, brought important but hard-to-find historical documents, including papers on computing by Babbage, Alan Turing, and John von Neumann, back into print. Finally, CBI administers the Tomash Fellowship, which supports one graduate student working on a dissertation topic in the history of computing each year.

The last of the three major institutions in the history of computing is the Computer History Museum.⁷ This holds an exceptional collection of rare and historical computer hardware, including pieces of the ENIAC, an Enigma machine, a SAGE console, a Cray 1 supercomputer, a Xerox Alto, an Altair, and an Apple 1. It also boasts large collections of photographs and software. Today, the museum is approaching its first anniversary in an attractive and spacious new home in Mountain View, California (part of Silicon Valley). Its exhibits are slowly taking shape, although a large amount of hardware is already on view with a minimum of explanation. The museum runs a high-profile speaker series and awards program, publishes a glossy newsletter, and maintains an attractive website. Like *Annals* and CBI, its origins can be traced back to the 1970s, though via a more circuitous path. It is a descendant of the now-defunct Computer Museum, once a successful attraction in downtown Boston. This earlier incarnation was closely tied to DEC, having begun life as the Digital Computer Museum in the lobby of a

⁶ <http://www.cbi.umn.edu>.

⁷ <http://www.computerhistory.org/>.

DEC building. By the 1990s, the leading local firms on which it relied for sponsorship were in terminal decline, and the museum closed. Much of its collection, however, ended up in storage in California, and became the nucleus of the new museum. The new museum is still taking shape, and so far at least has understandably focused primarily on fundraising and work with the volunteer community of computer enthusiasts, rather than on the support of historical research or the management of archival resources. It may evolve to become the primary interface between the history of computing community and the general public.

These are not the only institutions, of course. The Smithsonian includes several curators with an interest in the history of computing, and for the past decade has had an impressive array of hardware on display in its “Information Age” exhibit. It also holds several important collections of computing-related records in its archives. In recent years, a small group called the Software History Center⁸ has been active in organizing oral history events related to the history of the business software industry. In Britain, the National Archive for the History of Computing⁹ oversees records related to the history of British work in this field, while the Heinz-Nixdorf Museum¹⁰ in Paderborn, Germany displays an excellent collection in an extremely stylish setting. Most of the world’s major museums of science and technology, including the Science Museum¹¹ in London and the Deutsches Museum¹² in Munich, provide a reasonably extensive display of historic computers. There are also amateur groups scattered around that focus on the preservation and restoration of old hardware, and a number of smaller museums built around personal collections.

Since the emergence of its key institutions in the late 1970s, the history of computing community has gradually expanded. It has a reasonably coherent core, closely tied to *Annals*, CBI, and the Computer History Museum. This core group is quite small and quite chummy, in that we know each other well and tend to encounter each other once or twice a year at the *Annals* board meeting, related conferences, or special workshops or meetings sponsored by one group or another. If one combined the editorial board of *Annals* (which includes specialists from numerous universities in the United States and Europe, the Smithsonian, and CBI) with the staff of the other two institutions then one would have identified most of those who earn their living in the field, or focus their intellectual energies on it. There is, of course, a much larger group of people with a secondary interest in the field, including journalists and other writers who might be working on a book or article related to the history of a particular firm or individual, and numerous older people who lived through the history concerned and might contribute articles or personal memoirs on fields of particular interest.

Historians and the History of Computing

Professionals and Amateurs

The core group studying the history of computing is a mixture of eminent computer scientists and other old-time computing people with what one might be tempted to call professional historians (a tricky term, given that history is not a profession). However, “professionally trained historian” is a better defined category, and could reasonably be applied to anyone with an M.A. or Ph.D. degree from a reputable program in history or a specialized

⁸ <http://www.softwarehistory.org/>.

⁹ <http://www.chstm.man.ac.uk/nahc/>.

¹⁰ <http://www.hnf.de/>.

¹¹ <http://www.sciencemuseum.org.uk/>.

¹² http://www.deutsches-museum.de/e_index.htm.

history of science, technology, or business program. Most professionally trained historians active in the history of computing hold university positions (some in history programs, some in various “science, technology and society” programs, and some in information science schools), or work in museums as curators. A handful work as historical consultants, and a few more do historical work as a hobby while holding down more lucrative and unrelated day jobs with no historical content. There is also a shifting crowd of journalists, who usually wander into the field for a book or two and then wander off again to write about something different.

Over time, the balance between these groups has shifted toward the professionally trained historians, and this trend appears likely to continue in the future. The split is not absolute because many of the professionally trained historians studied or worked in computing before being trained in history. (This perverse career move at least provides them with one good joke to start their presentations with.) Professionally trained historians tend to take different approaches to their work from those favored by journalists or computing practitioners. They would tend to believe, fairly or not, that journalists are primarily interested in selling books by turning every story into a variation on the “X the lone genius who invented the Y and overcame decades of scoffing to revolutionize Z.” They would also complain that amateurs with a background in computing tend to be very good at amassing facts and details, particularly about computer hardware or events that they personally took part in, but aren’t very interested in addressing “broader historical questions” (by which they mean the kinds of things that historians talk to each other about).

Professionally trained historians, in contrast, tend to write in order to impress other professionally trained historians. You can’t blame them for this, since every academic discipline, including computer science, takes the same inward turn as an essential part of the development of a new academic field.¹³ One wants to publish work on hot topics, using new and exciting approaches. As a result, one gets hired by a good school and invited to the best conferences and workshops, and sees one’s articles cited frequently. Funding priorities also play an important role in computer science, and obviously less so in history. Combined with the immersion of graduate students within specific cultures, this steers individuals in particular directions and, on a broader level, shapes the field as a whole.

Indeed, historians feel a lot more guilty about this than most scientists, since they don’t use any equations; thus, they are haunted by the nagging sense that if they could just improve their turgid prose styles then maybe the public would be snapping up their books rather than the work of those journalists who craft huge presidential biographies, histories of the Civil War, and the recent spate of books about fishes, spice, or Jesus. This almost never happens, although with care an historian might reach a specialized non-academic audience of buffs.

How History Differs from Computer Science

The motivations and practices of academic historians have a lot in common with those you are familiar with in academic computer science. Both spend years in graduate school taking specialized courses and comprehensive exams before writing a dissertation and receiving a Ph.D. Both usually aspire to a tenure-track job at a good university, and pursue this goal by doing novel research, publishing in prestigious peer-reviewed journals, flattering their senior

¹³ In the early days of computer science, for example, theory and mathematical rigor were sought above all else by elite departments such as Stanford, in order to set aside computer science from mathematics on the one hand and service-oriented computer center work on the other.

colleagues, and accumulating passable teaching evaluations. Both trade material success for intellectual freedom.

Closer up, however, differences appear. Compared with typical computer scientists in the same institution, historians usually earn less money, teach more courses, and graduated from better schools with stronger resumes. This is a function of market forces because there are many more qualified historians for each advertised job, and because the only other significant source of professional employment for Ph.D. historians is museum work. Museum work doesn't exactly compete with Microsoft and Intel in pumping up salaries for top researchers.

There are no industrial or government research labs in history; in contrast, computer science has institutions like IBM Research, Bell Labs, Xerox PARC, Argonne, and Los Alamos. The reason is that the social motivations behind research in the humanities are fundamentally different from those in hard science and medicine. Society pays for research in medicine because it wants to cure the sick; at a lower level, it pays for research in computer science because of the clear benefits of its existing products such as the Internet, and because it might be helpful in finding new ways for the armed forces to more efficiently kill just the right people at minimum risk to themselves.¹⁴

On the other hand, society's main interest in research in the humanities is as a way of deciding which professors deserve to be hired or tenured and of encouraging them to stick around and teach more undergraduates. This makes more sense than you might think because an undergraduate degree in history, unlike a graduate one, is a very sensible major for an ambitious young person. It teaches students to read sources critically, assimilate large volumes of material of different kinds, write persuasive and factually based arguments, and reason with incomplete information. This is excellent preparation for law school, for Congress (Newt Gingrich claimed to be a history professor), and for even higher office. America's current leader has a degree in history from Yale, though I think it is fair to say the discipline isn't eager to take credit for his skills in the creative reading of intelligence sources.

Important historical research is carried out very differently from important research in computer science. A major research project in computer science is likely to receive hundreds of thousands of dollars in funding, employ several post-doctoral students, fill a lab or two with expensive equipment, provide dissertation topics for half a dozen Ph.D. students, and produce a long series of technical reports and papers, each with between three and six authors. It might well involve collaboration with other institutions or with industrial research labs.

A major historical research project, on the other hand, would typically be done by one person working for perhaps eight years. The result might be two long, painstakingly refined papers in good journals and one book of about 300 pages published by a leading university press. The major unit of scholarly production and exchange in history is the single-authored book, not the journal article, and certainly not the conference paper.¹⁵ Many of the most

¹⁴ The description of the social function of research in different fields is paraphrased from Robert X Cringley, *Accidental Empires* (New York, 1996). Historians footnote their borrowings religiously because failing to cite could be a career-ending mistake. This isn't to say that computer scientists are pawns of the military, and indeed they've proved very good at getting military money and then doing whatever it was they wanted to do anyway.

¹⁵ Few historical conferences even publish proceedings, and most presentations count for about as much as a book review when judging a vitae. The papers themselves are not usually reviewed prior to acceptance, though abstracts and resumes are. That is one of the most important differences between

important books in history began as Ph.D. dissertations, refined and improved for a few years after landing a first academic job. One book would usually be enough for tenure at a decent university, with two the benchmark for elite research-oriented institutions.¹⁶ Research grants are comparatively rare in history, and are usually for no more than a few tens of thousands of dollars—for example, to pay for a trip to an archive and a graduate research assistant for one semester. They are most likely to be received by historians working in fields like the history of medicine or business where well-endowed foundations have an interest in the field. There is no real equivalent for the role played by NSF and DARPA in computer science. The lack of research funding makes postdoctoral fellowships very rare in history, so even a successful young historian is likely to spend several years in visiting appointments with heavy teaching loads before landing a tenure-track job.

Historical research is thus a solitary kind of affair with little collaboration.¹⁷ One positive result of the lack of research funding is that, from the thesis on, the researcher has enormous flexibility in picking a topic to work on and in shaping the direction of the project. A student is not at the mercy of her advisor in being given a topic and hoping to receive reasonable billing on some of the group's papers. On the other hand, this leads to a much more detached relationship with less of an investment in the student's work. Historians, ironically, never bother to prepare elaborate academic genealogies of the kind popular in theoretical computer science.

History job announcements rarely specify an interest in the history of computing. This means that a hopeful researcher in the area instead has to market herself as a special case of an historian of something else, who happens to be interested in computer-related topics. The problem is that the people doing the hiring will almost certainly know nothing about the history of computing, and will probably see it as peripheral to the researcher's main area of interest. I am not personally aware of any case in United States in which an individual has written a dissertation focused primarily on any aspect of the history of computing, and then been hired by a different institution for a tenure-track job including research and at least some teaching in the history of computing.¹⁸ That surely must be the minimum standard by which to claim that an area has begun to function as an accepted disciplinary specialization.

history and computer science (or indeed science in general), where books are usually textbooks and count for very little when it comes to tenure or hiring decisions.

¹⁶ I should also make clear that, unlike textbooks in computer science, these history books are no more likely to make appreciable sums of money for their authors than are journal articles. They are purchased primarily by libraries, and a book selling several thousand copies would be considered an enormous success. Advances would not usually cover the Xeroxing expenses, let alone the author's time. The books are effectively subsidized by the university employing the author or, given the teaching loads common in the humanities, are produced in the author's evenings and weekends. So here, too, "pull" from possible purchasers counts for a lot less than the "push" provided by tenure committees. The fact that the public has an insatiable appetite for books about Nazis, presidents, and wars has therefore not made these topics particularly attractive to professional historians. (This is analogous to the disregard shown by computer science programs to the stated priorities of industrial recruiters.)

¹⁷ Jointly authored books and papers are by no means unknown in history. This point was brought up in discussion at one of the workshops. In response, William Aspray, who has worked on numerous joint projects, pointed out that while this can be a way of getting additional perspectives into a piece of research, it might actually prove to be more work to write an historical piece jointly than to handle it individually. This is because history is so tied up with writing, which is hard to divide up into self-contained activities.

¹⁸ Things are not quite as bad as this suggests. There are knowledgeable people currently teaching and publishing on the history of computing in the United States, but they were hired to do other things or stayed on at their Ph.D.-granting institution. In other cases, people with specialist training in the history of computing found jobs in other fields (such as information science, science and technology studies, and

As peripheral as the position of historians of computing is in historical fields, they have proved to be much more welcoming than computer science departments. In professional fields such as medicine, library and information science, and even business, a handful of leading schools have hired historians and added history courses to their core curriculum. This has never happened in any American department of computer science, and there are no signs that it is likely to happen any time soon.

Different Kinds of History

So far I've talked about history as if it was a single field. In fact, history is a highly segmented academic discipline. Many job announcements in computer science state several general research areas of interest, and hint that exceptional candidates in other areas will be considered. In contrast, history jobs usually stipulate a geographical region, a time period, and a methodological approach. A school might seek a historian focused on the United States in the second half of the nineteenth century, working on issues of business, labor, and social history. Or it might emphasize Western Europe during the Renaissance, looking at the history of the physical sciences. One recent opening even specified expertise in the history of Delaware, among a long list of other requirements. The job market is such that search committees rarely have to compromise.¹⁹

The point of telling you all this is not to fill you with sympathy for the plight of the historian, but rather to give you a sense of why the history of computing has developed so differently from research areas within computer science, where the emergence of SIGs, journals, influential conferences, and influential centers of research has often been comparatively smooth. Because research funding is scarce and collaborative research unusual, research progresses fitfully and according to the whims and interests of the individual researcher. There are no tenured, well-funded professors with armies of graduate students ready to fan out and probe different aspects of the field. Although there have been a few attempts to convene the great and good to identify future research agendas for this field, these have been hindered by the lack of researchers to direct and by the fact that the history of computing community cannot collectively hire, fund, or tenure researchers. So research priorities rest not just on the whims of individual professors and graduate students, but on the whims of individual professors and graduate students who have to meet the standards of a wide range of different fields.

Some of these weaknesses are also strengths. One positive result of the proliferation of historical specialties, coupled with the non-existence of the history of computing as a recognized academic specialty, is that those who do choose to work on topics in this area bring with them the techniques and concerns of many different historical fields. As historians gradually come to grips with the events of the past half-century, many more historians will surely recognize the

informatics) in which no historical teaching was possible, but have continued to research and publish on historical topics.

¹⁹ Overall, one of the most important factors shaping research directions has been the kinds of people hired to academic jobs, which in turn has been shaped by the kinds of courses taught to undergraduates. In the 1960s and 1970s, expansion in university education and changing social attitudes sparked a turn toward social history, and an examination of representative experiences rather than studies of elites. Examination of class, race, and gender became the central concern of historians. More recently, this has been supplemented by attention to cultural history, and the study of things like popular films and magazines. Much hiring is taking place in identity-fields such as black history, Jewish history, gay and lesbian history, and so on.

importance of computer technology to crucial developments in science, business, culture, and daily life.²⁰

Here are some of the main historical subdisciplines with clear conceptual connections to the history of computing:

History of Science: History of science is quite well established as an academic field. There are a number of free-standing history of science departments in leading American universities such as Harvard and the University of California, San Diego, and there is a tendency to include a lone historian of science in many of the larger departments of history. It runs from pre-history onwards, with some scholars focusing on classical science and many on the scientific revolution and the enlightenment. Historians of science usually specialize in geographic areas and historical time periods, such as ancient China or twentieth-century Latin America. Many interdisciplinary programs in “science studies” or “science, technology and society” at universities such as Cornell include strong historical dimensions. The major society in this area is the History of Science Society, whose journal, *IS/S*, is the leading American publication of its kind.

Computer technology plays an increasingly vital set of roles in almost all scientific disciplines. Examples include data reduction and experimental control in physics, modeling efforts in meteorology and economics, gene sequence reconstruction, statistical analysis, scholarly collaboration, and information dissemination and retrieval. Computers have changed the laboratory practices, publication patterns, analytical methods, and standards of evidence and proof in many fields. In some areas they have provided entirely new avenues for experimental work. As early as the 1960s, computing was radically reshaping disciplines such as numerical analysis. Few of these topics have received any attention yet from historians of science, but this will surely change. In addition to scientific computing, the other big topic for historians of science should be the history of computer science itself. Computer science, and the related fields of information science and software engineering, were among the most important disciplines to emerge in the past half-century, and have begun to attract some historical work (so far primarily by members of the disciplines concerned).²¹

History of Technology: As a field, the history of technology is a little smaller and less well established than the history of science, but may be catching up. In America, its roots lie primarily in the history of engineering (largely practiced, for obvious reasons, in schools with strong engineering programs) and in the efforts of curators at the Smithsonian and other museums. The Society for the History of Technology has almost two thousand individual members, and publishes *Technology of Culture*, the field’s leading journal. Historians of technology are paying increasing attention to the use and consumption of technology as well as its design and production, and are expanding the range of technologies they study beyond large, metal things such as railways and space rockets and toward things like zippers, hair products, and personal stereos. Much work in recent decades has been framed as an examination of the processes by which technologies are shaped by social processes and

²⁰ As a rule of thumb, it takes about fifty years after the events concerned for historical scholarship on a particular area to really mature to the point at which historians have come to understand the events of a particular era in ways different from those held by the people who lived through it. This achievement of “historical distance” takes place partly through the death of those concerned, who tend to have strong and partisan notions of what happened.

²¹ See the sources cited in “Key Resources in the History of Computing” (chapter 26, this volume), under the heading *Computer Science*.

interests, but in turn reshape society through their use. Historians of technology have so far paid rather more attention to computing than have historians of science. Most work thus far has addressed mainframes, minicomputers, and supercomputers, and has paid particular attention to the evolution of the computer from earlier technologies and to the roles of government and business organizations in steering that evolution. However, it seems certain that, for the 1970s onward, the proliferation of personal and embedded systems in areas such as cars, home systems, PDAs, cellular phones, navigation systems, domestic appliances, audio and video devices, and Internet communications will move computer-related topics toward the heart of the history of technology.

Business History: Business historians are found in both history departments and business schools. The subject is fairly marginal in both kinds of school, with a few notable exceptions such as the Harvard Business School. Traditionally, much work in business history has consisted of celebratory corporate histories commissioned to mark major anniversaries. Successful or prominent firms, such as Microsoft, often attract hastily written books by journalists looking to cash in on the huge market for easy-to-read business stories. However, academic historians often produce more analytical and critical work, and have been paying increasing attention to the evolution of business institutions (such as the multinational corporation), the development of managerial specialties, the creation of new products and industries, and the relationship between corporate structures, markets, and customers. The leading American society in this area is the Business History Conference, which now publishes a journal called *Enterprise and Society*. This competes with Harvard's longer-established *Business History Review*. A considerable amount of work has been published on the business history of the computer industry, including a shelfful of books on IBM and studies of the other early computer companies. Professional historians have done relatively little work on the prominent firms of the microcomputer and Internet era, in part because none have made their archives available.²²

Separate from the business history of computer firms, and so far much less well developed, is a literature on the use of computer technology in different kinds of organizations. Computers have long played a vital role in business administration. More recently, firms have redesigned their core processes and activities in new ways made possible by computer technology. In addition, computers may finally be living up to their promise of reshaping organizational structures, eliminating many kinds of middle management positions, and supporting managerial decision making through the integration of knowledge and data from different parts of the firm. Corporations rely on a huge new class of expert technicians to keep their operations running. While few business historians have yet written about these changes, they will be very hard to ignore.²³

Economic History: Economic history is not entirely distinct from business history, though the division has been growing wider in recent years. Economic historians tend to be trained as economists, and apply modern quantitative methods to data sets gathered from historical

²² Pretty much every computer firm of any note has attracted at least one history, most of them written by journalists (around the peak of the company's fortunes) or participants (often long after the firm in question vanished). Professionally trained historians have written a number of business histories, notable among which is Martin Campbell-Kelly, *ICL: A Technical and Business History* (New York, 1989).

²³ A number of works on the use of computers in business by Thomas Haigh, JoAnne Yates, and James Cortada are cited in the resource guide. Another example, produced by a Harvard Business School team, is James L. McKenney, Duncan C. Copeland, and Richard O. Mason, *Waves of Change: Business Evolution through Information Technology* (Boston, MA, 1995).

sources. The main organization for economic historians is the Economic History Association. Much economic history concerns fairly technical topics such as the comparative history of central banking in different countries, or trade patterns in the classical world. Economists are sometimes attracted to historical examples as a way of testing broader theories, which often leads to repeated reexamination of particular topics from competing viewpoints while other topics go unexplored. For example, economists have looked at earlier waves of technological investment in railroads and electrical power to compare their magnitude, returns, and duration with the recent wave of spending on computer technology. They have also argued about whether the continuing dominance of the QWERTY keyboard layout (famously created to slow typists down) proves that historical accidents can lock society into suboptimal technologies (a process known as “path dependence.”) Other computer-related topics include the management of high technology innovation, and the potential role of government in protecting and encouraging nascent technological industries.²⁴

Other Areas: Many other historical specialties can, or should, interpret aspects of the history of computing according to their own interests. For example, a few decades ago a number of prominent labor historians and sociologists developed an interest in the role of computerized automation in transforming skilled and manual work in areas such as factory work, banking, and clerical labor.²⁵ Oddly, research in this area has declined sharply since the 1980s, but an eventual resurgence seems likely. Many historians study the role of government and its many agencies, departments, and programs, but so far have paid little attention to the role of computers and punched card machines in shaping its development.²⁶ However, quite a lot has been written about the role of government funding in the development of computer technologies.

How might these fragmentary perspectives prove to be an asset? Consider, for example, the history of operating systems. Different groups might see it in very different ways. An historian of science might focus on the computer science side of the story, looking at Dijkstra and his conceptual work on semaphores, or use the commercialization of time-sharing as an example of transfer of skills and ideas from academic researchers to industrial groups. An historian of technology might be more interested in the role of System/360 and its much publicized problems in the development of software engineering as an academic field. A business historian, on the other hand, would be much more interested in the history of Microsoft or Novell as corporations, looking at marketing techniques, technological strategies, management methods, and so on. Or, focusing more on applications, the business historian might decide to examine the role of operating system developments in facilitating the implementation of real-time, database-oriented applications for management and control purposes. A labor historian, in contrast, might see something completely different. The term “operating system” was coined because it was supposed to replace the work of computer operators (themselves often retrained punched card machine operators). Over decades, it shifted a large amount of work between operators and application programmers, and created a new specialty of systems programmers. An historian interested in the history of government

²⁴ One example of an economic history approach to a topic in the history of computing is David Mowery, *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure* (New York, NY, 1995).

²⁵ Among the most notable of these were Harley Shaiken, *Work Transformed: Automation and Labor in the Computer Age* (Lexington, MA, 1986); and Shoshana Zuboff, *In The Age of the Smart Machine: The Future of Work and Power* (New York, NY, 1988).

²⁶ A notable work addressing the history of computing and other information technologies in the development of the British government has recently appeared, and may spark some interest in the topic. Jon Agar, *The Government Machine* (Cambridge, MA, 2003).

support for research and development might focus on the role of military projects in supporting developments, including the SAGE project of the 1950s and ARPA funding of time-sharing.

Historical Research Methodologies

As diverse as these different kinds of history are, there are still some fairly safe generalizations that can be made across them. For the computer scientist, the most important one to grasp is that historians do not see themselves as practicing a science, even a social science. This has profound effects on the structure and practice of historical research. Social sciences such as sociology, economics, and specialist fields such as information science have become increasingly mathematical and quantitative in a quest for scientific rigor. Researchers in these areas win acclaim by gathering precisely controlled data sets, framing and testing extremely narrow research questions, and achieving results with a high level of statistical significance. Their research papers include long and explicit discussion of methodology, and segregate descriptive presentations of results from analysis and analysis from conclusions. They embrace the turgid language, passive voice, and specialist vocabularies beloved of the harder sciences.

Historians, in contrast, see themselves as practicing a craft. Their job is to take the mass of written documents and other sources (such as artifacts and audio recordings) preserved from the past and use them to tell stories or, as they would put it, to craft historical narratives. Social or hard scientists sometimes pick up the work of historians and frown in bafflement at its apparent lack of rigor. After a brief introduction, an historical work usually tells a story, generally in historical order. In a longer work, each chapter tells a self-contained part of the larger story. There is no lengthy framing of research questions, no detailed defense of data-gathering and analytical methodology, and no long presentation of conclusions and recommendations. No historian bothers to ritually cite seminal papers by the founding fathers of the discipline or exhaustively survey all previous work in the field. The result, our hypothetical scientists might feel, is just a descriptive presentation of raw data, lacking entirely in theory, analysis, or intellectual framework. History work would, in the famous phrase, consist of nothing more than writing down one damn thing after another until the book was full.

That would be a fundamental misreading of the historian's craft. During their graduate training, historians spend a great deal of time learning research methods, critiquing the work of others, discussing the strengths and weaknesses of different approaches, and practicing different techniques. Any good historian puts an enormous amount of thought into framing a coherent and meaningful research structure, defining historically accurate and analytically useful categories, and tying the specific story being told into broader issues, such as the development of the middle class or the relationship between race and gender. The more fundamental difference is that social scientists make a ritual and energetic display of all this work in the final product, whereas historians are trained to hide the evidence of their labor to leave the final narrative as unencumbered as possible. I like to think of this as being akin to the work of a sculptor who uses many kinds of molds, scaffolds, and sketches during his labors, but disassembles and destroys these as the work progresses, creating a final product that gives an impression of effortless verisimilitude to life. The social sciences, on the other hand, are more like postmodern buildings, such as the Pompidou Center in Paris, which display their internal workings on the outside so as to draw attention to them.

Historical research is inseparable from the telling of a particular story. As Michael Mahoney noted during discussion of this point at the conference, a computer scientist who says that the research is finished and just needs to be written up is indeed nearing the end of a

project. An historian who says the same thing has not really started. Indeed, one of the most important pieces of advice given to graduate students in the field is to write up material from the very beginning to see how the story hangs together and what additional material is needed to tell it. Without doing this, one might accumulate thousands of pages of notes that will never coalesce into a coherent project. Because of this focus on narrative, historians tend to be better writers and tend to take teaching more seriously than do computer scientists. There are, of course, many exceptions to this generalization on both sides.

Theory is not a prestigious area within history. Historians do sometimes produce articles urging more attention to some particular approach or topic, and journals sometimes publish articles reviewing existing work in an area, categorizing its strengths and weaknesses and setting out a possible agenda for future work. They call this “historiography.” In general, however, the editors and reviewers of historical manuscripts like to keep such discussion to the minimum necessary to show some kind of relevance to earlier work and imply that a novel and important contribution of some kind is being made. Historians are deeply suspicious of anything more abstract, and are usually amused more than threatened by occasional efforts to produce grand unifying theories of historical change, or to apply cybernetics or mathematics to the discipline.²⁷

Historians decry “presentism”—the tendency to interpret historical events and ideas in terms of our present-day interests and assumptions, rather than doing the work necessary to understand what they meant to the people involved. For example, the team responsible for the database query language SQL originally believed that they were producing a tool for non-technical managers to directly query databases using an easy-to-grasp subset of English. Only much later did they realize that its main use was as a lingua franca for application programs to communicate with databases and for the transfer of data between databases. It did not solve the problem it was created to deal with, but thrived anyway as the solution to a different one. One can’t understand why SQL took the form it did without appreciating this.

A related practice is “Whig History,” which is a story consisting entirely of deliberate and unbroken progress from primitive beginnings to current perfection. In this view, the only important things in history are the “contributions” made at various points to our current understanding of the world. The history of computing is especially vulnerable to these problems because the real, sustained, and rapid progress made in many areas of hardware and software development does tempt one to view the past as a simple march of progress. One problem with this view is that a history consisting only of progress towards a fixed goal has erased the most valuable lessons we might learn from it on topics such as the need for flexibility, the difficulty of foreseeing the future, ideas that were bad then and shouldn’t be tried again now, and even ideas that were bad then but might work now.

While rejecting scientific methodologies for their work, historians do display a profound, almost obsessive, attachment to certain kinds of rigor. The methodology banished from the text is displayed instead in the footnotes, which might (at least before editors go to work) comprise a full one-third of the total length of manuscript. While explicit discussion of methodology does sometimes occur here, it is more important for them simply to list the source of every quotation

²⁷ A generation ago, in the late 1970s and early 1980s, there was a push toward extensive use of statistics and databases in history, but these approaches have almost vanished from mainstream historical training. They survive only in interdisciplinary areas such as historical demography and economic history, but these fields have been left almost entirely to those trained as demographers and economists.

or fact presented by the author. The skilled reader is able to essentially reverse-engineer the author's methodology from these citations. Indeed, faced with a new work in his own field the historian frequently turns first to the footnotes, scanning them to determine the range of sources, quality of archives, number of different collections, geographical range, social diversity, and general heft of the evidence upon which the author has drawn. A paper about a famous part of computer history, say the development and use of VisiCalc (the first spreadsheet) based largely on existing popular histories and reading of a single newspaper, will impress much less than one based on corporate records, hitherto unknown personal correspondence, laboriously cross-referenced consultation of computer dealer trade journals, user group newsletters, popular computing magazines, and interviews with surviving users, developers, and dealers.²⁸

This stress on using a diverse range of sources to present a balanced picture means that historians rarely espouse a doctrinaire commitment to one or another theoretical framework. They are often pragmatists at heart. Most historians eschew the flamboyant postmodernism popular in departments of literature. Thoughtful historians have always known that there can be no neutral, complete, and empirical description of the past, and that any historical narrative must make assumptions, stress some things over others, and even favor the perspectives of one group over another. Each generation inevitably reinterprets the past in the light of its current preoccupations, which is why medieval and classical historians still have useful work to do. Yet historians do not conclude from this that it is unnecessary to strive for honesty and fairness in their craft, or that it is futile to pile footnote upon footnote, or that all possible accounts are equally valid. Indeed, sins such as the failure to properly cite quoted half-sentences, or an apparently deliberate misrepresentation of archival evidence, have led to the disgrace of more than one prominent historian in recent years. It may be that historians are too extreme in their erasure of methodology, too skeptical of big ideas, and too stodgy in their reliance on footnotes, but theirs is, by and large, a noble and unpretentious calling.

Topics Covered and Not Covered

Given the smallness and diversity of the history of computing community, it is not surprising that its coverage of important historical topics has been far from uniform. Scholars working on some historical topics, such as the Civil War (British or American) or the politics of the New Deal, must do a lot of work to identify a topic not already written about, or to show that their insights render inadequate a shelf of existing books on the topic. In the history of computing, however, almost no topics are over-studied. The only three exceptions that spring to mind are journalistic profiles of Bill Gates, arguments about who really invented the computer, and discussions of the ENIAC.

Several attempts have been made to write overall histories of computing. Their scope has broadened over time, from histories of calculating devices (a topic that historians came to understand quite well) to more ambitious attempts to present the computer as an evolution of existing technology in fields such as business administration. This, of course, has been a delayed reflection of the changing use of computer technology. The problem, however, is that now that computers are replacing cameras, telephones, record players, and postal systems it is becoming impossible to attempt to write a comprehensive history of computing that covers all these areas, examines applications and users (which many professionally trained historians of

²⁸ Of course, in many cases there may not be any archives to draw on, which highlights the importance of historical preservation and explains why the same institutions (e.g., MIT and Metropolitan Life) keep coming up again and again in historical work: historians write about the material found in archives that is well cared for and accessible, and tend to ignore the material that isn't.

technology believe to be vital), and puts the computer into context as an extension of what went before.

Academically trained historians of computing tend to agree that a shift in emphasis from study of the history of hardware to the history of software is intellectually desirable, given the increasing economic and technical importance of software and its vital role in determining the actual social functioning of computer systems. Unfortunately software is much harder to study. Software is less likely to be preserved than hardware; it is produced by many, many times more groups (mostly in end-user organizations); there are fundamental differences among system software, application software, and scientific software; and it is often impossible to separate software from business services or processes. Most attention given so far to the history of software by professionally trained historians has focused on attempts to formalize coding practices through the use of formal methods and the creation of software engineering as a true discipline.²⁹ These topics loomed large in the minds of credentialed researchers, but do not appear to have been particularly influential on most actual programmers; therefore questions of what ordinary programmers actually did remain almost entirely unaddressed.

Some areas, of course, have been explored much better than others. ACM SIGPLAN organized two major conferences on the history of programming languages, including lengthy articles by the creators of key languages, and discussion of their design and subsequent evolution. These were published in book form.³⁰ Similar efforts were mounted by those involved by the designers (and in some cases early users) of many important commercial computers such as the IBM 701, IBM 650, and Burroughs 5000, resulting in special issues of *Annals of the History of Computing*. The history of early firms involved in the computer business has been quite well covered, including a remarkably detailed series of technical histories of IBM's work in the 1950s and 1960s. The stories of the first, one-off experimental computers such as the ENIAC, EDSAC, and WHIRLWIND are quite well documented. Much work in these areas consists of the memoirs of participants or of very detailed and narrowly focused accounts of the development of a specific system. While valuable and interesting in their own right, many of these topics are still waiting for professionally trained historians to revisit them and put them into a broader context.

The history of many other important technologies, including operating systems and database management systems, has gone almost undocumented. Indeed, despite the popularity of Microsoft and Bill Gates as journalistic subjects, nobody appears to have tried writing an overview of the history of operating systems, whether scholarly or popular. There are plenty of fun-to-read, journalistic books about the early history of the personal computer industry, including at least one decent-sized shelf devoted entirely to Apple. So far, however, no

²⁹ The body of work on the intellectual development of software engineering concepts includes Michael S. Mahoney, "Finding a History of Software Engineering," *IEEE Annals of the History of Computing* 25/1 (Jan-Mar 2004), <http://www.princeton.edu/~mike/articles/finding/finding.html>; Donald MacKenzie, *Mechanizing Proof* (Cambridge, MA, 2001); Stuart Shapiro, "Splitting the Difference: The Historical Necessity of Synthesis in Software Engineering," *IEEE Annals of the History of Computing* 19/1 (Jan-Mar 1997): 20-54; James E. Tomayko, "Software as Engineering," in *Mapping the History of Computing: Software Issues*, ed. Ulf Hashagen, Reinhard Keil-Slawik and Arthur L. Norberg (New York, 2002); Nathan Ensmenger, "From Black Art to Industrial Discipline: The Software Crisis and the Management of Programmers," Ph.D. dissertation (University of Pennsylvania, 2001); and Maria Eloina Pelaez Valdez, "A Gift From Pandora's Box: The Software Crisis," Ph.D. dissertation (University of Edinburgh, 1988).

³⁰ Richard L Wexelblat (ed.), *History of Programming Languages* (New York, 1981); and Thomas J Bergin and Rick G Gibson (eds.), *History of Programming Languages II* (New York, 1996).

professionally trained historian has ventured more than a chapter on the personal computer era, which has done little to advance our understanding.³¹

Almost nothing has been written about the history of computer science as an academic discipline. The main exceptions are a series of articles on early computer science departments and NSF funding by William Aspray, some short articles on theoretical computer science by Michael Mahoney, and two books on the role of DARPA in funding computer science. There are many different ways in which the topic could be approached, including a conceptual examination of the influx of ideas from different areas of mathematics, institutional histories of key departments, books on the development of ACM and the other relevant academic societies, or biographies of key computer scientists. The ACM has recently made some moves toward support of historical work, which if successful will help in preserving its own history and that of the discipline.

The history of scientific computing is reasonably well served into the mid-1950s, at which point the secondary literature largely gives out. There have been a number of publications covering table-making in the USA and UK and differential analyzers; a flurry of material on Charles Babbage; a good biography of von Neumann; and material on the service organization created by IBM to support its first scientific users. With the exception of one paper on supercomputing and the national labs, the only papers that come to mind for the post-1955 period are retrospectives of major machines and packages written by the creators themselves, often to celebrate the 20th or 25th anniversaries of their introduction.³²

In short, then, the queue of worthy topics is continually growing longer, as new areas of study develop and pile up more rapidly than the small but doughty band of historians can deal with the old ones. Whatever your personal area of interest, you'll probably find that almost all of it has so far escaped the notice of historians, though you will probably find some interesting and insightful papers on related topics. However, to look on the bright side, an impressive quantity of excellent work has appeared in the last few years, and there are signs that historians are starting to gain an intellectual purchase on some of the foothills of this imposing mountain range.

What to Learn from History

Historian Daniel Boorstein famously wrote that planning for the future without an awareness of the past is like trying to plant cut flowers. This view finds considerable support among historians, who quote it at every opportunity, and very little among technologists. In this they are not so very different from the mainstream of American culture. As a people, white Americans have an unusually abstract relationship to their own history. It's not that Americans downplay history. A pervasive, state-fostered, and stylized mythology of turkey-eating pilgrims, brilliant founding fathers, selfless flag-sewing women, brave guerilla fighters, and ingenious

³¹ Historians are just beginning to come to grips with the post-1975 period. The best example so far is probably the discussion of the microcomputer software industry in Martin Campbell-Kelly, *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry* (Cambridge, MA, 2003).

³² See the "Scientific Computing" section of my "Key Resources in the History of Computing" for citations and details of the books mentioned here. Not included there are the paper on IBM's first scientific support group by Atsushi Akera, "IBM's Early Adaptation to Cold War Markets: Cuthbert Hurd and His Applied Science Field Men," *Business History Review* 76/4 (Winter 2002): 767-802; and the paper on supercomputing by Donald MacKenzie, "The Influence of Los Alamos and Livermore National Laboratories on the Development of Supercomputing," *IEEE Annals of the History of Computing* 13/2 (April 1991):179-201.

inventors serves to rally patriotic sentiment on national holidays, in preparation for clearance events at the local mall. But while the flag and the constitution retain enormous symbolic power, they resonate very much as symbols of present rather than past glory. In many other places such as Ireland, Greece, or much of the Muslim world, discussion of current grievances and disputes inevitably blends with reference to events taking place hundreds or thousands of years ago. This is not all bad. As an Englishman, I find it reassuring that I can safely attend a 4th of July barbecue or walk across the famous bridge at Concord without anyone feeling any upwelling of “ancient hatreds” toward me. Professionally, however, this pervasive indifference to the substance of the past makes the teaching of any kind of history to non-specialists an uphill struggle.

The Denial of the Past

A few years ago, amid the frenzied hype of the Internet boom, one of the long-time columnists of one of America’s most successful personal computing magazines addressed the issue of technological change. He reflected that the wave of technological change brought about by the Internet was entirely unprecedented, and that his own age faced challenges and opportunities unlike any other. His great-grandfather, he suggested, had died in a world very much like the one he was born into. He, on the other hand, had lived through upheavals of ever increasing magnitude. Similar claims are made all the time, but one needs only a very passing acquaintance with history to realize that this is complete rubbish. Imagine an American who was born in 1895 and died in 1970. She would have seen, among many other things, the development of personal transportation from the biological horse to the Ford Mustang, the development of communication from the telegraph to the communications satellite, the development of home entertainment from the player piano to the stereo LP, the development of the film studios, talkies and Technicolor, the creation of television and radio broadcasting and networks, the introduction of electrical refrigerators, the Apollo landings, nylon, antibiotics, the atomic bomb, intercontinental passenger jet service, supermarkets, suburbs and shopping malls, the mainframe and the minicomputer.

Admittedly the half lifetime since 1970 has replaced the LP with the CD; the Boeing 747-100 with, well, the Boeing 747-400; and big malls with enormous malls. It has invented the SUV, and has seen huge drops in the cost of long-distance communication and incremental advances in medical care. But the fact that such a claim could go unchallenged in one of America’s best-selling technological publications indicates a depressing triumph of ignorance. If people had been just a little less ignorant of history, they might not have believed that, for example, a grocery home delivery firm like Webvan could be the next Microsoft. For one thing, they would have known that grocery stores used to make deliveries, and that the practice died out with the rise of large supermarkets, increasing population diffusion, and greatly increased ownership of personal automobiles during the first half of the twentieth century. The web browser didn’t change those things. History is sometimes defined as the study of change over time, but paradoxically the most valuable thing history can contribute to many areas is an appreciation of continuity.

The computer field is divorced from its own past. Programmers can be viewed as “obsolete” at the age of thirty-five. In the popular view of the past forty years, one technological transition created the computing world we know (always about five years ago) and another one will arrive to fix all the problems with it (sometime next month). It was hard to convince PC programmers that they had much to learn from the software engineering techniques developed for mainframe projects, and it is hard to convince young Java programmers that any of the problems they grapple with might have existed ten years ago. But we know that, on the

technical level at least, every idea, every system (or at least all those in real use), and every architecture is constantly evolving. Without a knowledge of history, we can never understand why things are the way they are.

This lesson is a hard one to convey to students. In one recent class I assigned for my students an excerpt from Ellen Ullman's book, *Close to the Machine*, in which she spoke of the dozens of now-obsolete languages she had used over the course of her career and pondered the mass of soon-to-be-obsolete material arriving with every new shipment from her Microsoft Developer's Library subscription.³³ In a discussion section, the students were asked to discuss their own feelings about entering a field in which they would be required to relearn their entire technical skill-set on a frequent basis. Even after reading the article, few of them believed that this would happen to them. They were confident that the smattering of Java and Windows they had picked up during their education would stay current indefinitely.

Knowledge of the broader history of technology, and indeed of history in general, may be constructive here. History may also help the discipline of computer science—and the related fields of information science, information systems, informatics, and software engineering—figure out what they are. For almost forty years, employers have been complaining that universities produce computer science graduates with bad attitudes and a dearth of useful skills. People in universities have countered with the claim that they give education rather than vocational training, and provide the core skills and understanding that underlie current and future practices. That's how they justify having spent decades teaching Pascal, SML, LISP, and Smalltalk rather than COBOL, Visual Basic, and PERL.

Academics are thus charged to recognize the "timeless principles" in their own fields, knowledge of which will benefit students for many years and through changes in specific technologies. It seems reasonable to suggest that having knowledge of what has remained stable in the past may help us to work out what is liable to remain stable in the future. In my own research, for example, I discovered that many central dynamics of corporate computer use had remained unchanged over its fifty-year history, such as the difficulties of diagramming and describing business processes, the refusal of most developers to use the methodologies favored by academic experts, and the inescapable fact that most computer departments are hated by their users.³⁴ And, I would suggest, the same is true of the computer science field itself, and of all its various subdisciplines. To know who you are, you must know where we came from. You must know your own story. Why is computer science what it is, and not something else entirely? When the time comes to shift, how will you know? What should you keep, and what will you discard?

Well, enough of the rhetoric and hand-waving. I would suggest, however, that perhaps the biggest contribution that history can make to improving undergraduate education is in improving the understanding that undergraduate educators have of their own field and where it came from. It might never need to show up on the syllabus at all to accomplish that.

Tactical and Strategic Users of History

Let's assume, though, that an instructor or curriculum designer is convinced of the utility of including historical material in a computing course. I'll leave it to the other contributors to talk

³³ Ellen Ullman, *Close to the Machine: Technophilia and its Discontents* (San Francisco, 1997).

³⁴ Thomas Haigh, "Technology, Information and Power: Managerial Technicians in Corporate America," Ph.D. dissertation, University of Pennsylvania (Philadelphia, 2003).

about exactly how that might be accomplished. I do, however, believe that there is one vital question here, and that each contributor has implicitly come down on one side or another on this question, without necessarily realizing this consciously. That question is whether the contribution of history to computing education should be as a tactical tool designed to help students grasp the same material and build the same skills that they are already receiving, or as a strategic challenge to fundamentally rethink the curriculum design, skills, and culture of computer science education.

One can imagine history playing a useful part on the tactical level either as a pedagogical innovation comparable to the use of PowerPoint, in tutorial sessions, or in on-line discussion groups. It's not hard, for example, to believe that an understanding of the historical development of file management systems and the network data model might help students to appreciate the strengths and weaknesses of the current DBMS systems and of the relational model. Dropping snippets of biography into the class, to let students know who Turing was when they discuss his machine, might help to keep them awake. This might not involve what most academic historians would think of as "good" history. The history would often consist of little, disconnected snippets of factual material, stripped of its richness and complexity, its contingency and context. Instructors might hide details and skip over complexities to create neat parables or inspirational fables, such as the apocryphal but widely repeated story that the former chairman of IBM stated that the world market for computers would be around five, or the story that NASA spent millions to create a space pen while the Russians used pencils. On the other hand, if it helps to teach good computer science, who cares if it is bad history? Purists in any area tend to be shocked at what the masses do with their work.

There are also some areas of computer science where integration of historical material is an obvious tactic, perhaps an inescapable necessity. For example, in a computer architecture or operating course it is hard to begin with discussion of current systems because they are enormously complex. Many instructors would start with simple, manageable systems (such as Andrew Tanenbaum's MINIX, an instructional recreation of early UNIX), perhaps recapitulating during the progression of their courses some of the same evolution toward increasingly complex systems that has taken place over history. Perhaps better history might help to produce better texts, stressing the specific factors influencing different designs, just as business schools use detailed case studies to teach hard-to-formalize kinds of skill.³⁵ Likewise, theoretical computer science might follow mathematics in presenting a carefully ordered sequence of developments in which each new topic rests on the proofs presented earlier in the course, and so resembles a rather stylized and formal kind of history.

How about the idea that incorporating history in the teaching of computing would fundamentally shake up our assumptions about what it is that students should be learning? This position appeared to be held, consciously or subconsciously, by almost all the professionally trained historians at the workshop and by none of the practicing computer scientists. As one historian put it, "We should be training them to think like historians." What might this mean in practice?

It is important to realize that undergraduates are not usually taught history because anybody believes that the factual content of a course on, for example, nineteenth century

³⁵ MINIX is described in Andrew S. Tanenbaum and Albert S. Woodhull, *Operating Systems: Design and Implementation, 2nd Edition* (Englewood Cliffs, 1997). An important operating systems text with a developmental perspective is Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne, *Operating System Concepts, 5th Edition* (New York, 1998).

diplomatic history will be directly useful to them in their work lives. There are certainly specific lessons there for policymakers dealing with current challenges, but history is not marketed primarily as vocational training for future State Department officials. The most crucial skill for a history major to develop is not necessarily the craft of tracking down obscure historical sources, but rather that of interpreting the biases and assumptions of their authors and fitting them together into a coherent whole.

For example, in 1998, and again in 2001, I taught a course on the idea of computer technology and social revolution, designed to help students get a handle on the overblown claims being made for the social impacts of the Internet by looking at previous technologies, particularly the mainframe in the 1950s and the PC in the 1970s and 80s, but also apparently unrelated technologies such as nuclear power, ham radio, and the telegraph. Students examined these earlier technologies, and saw many kinds of similarity with current ones (from the utopian claims made for them, to the hobbyist cultures that developed around them, to the alarmist warnings on their threats to traditional values). Another course mixed historical and recent examinations of the role of information in structuring companies, markets, and customer relationships in business. These courses gave the students a perspective to deal more critically with technological hype, not just for dot-com stocks (which vanished without any real help from me), but with future waves of technological mania as yet undreamt of.³⁶

In a seminar course of this kind, intensive group discussion of source material is vital. The source material is not in itself the most important thing. In my teaching I've assigned the film *Wargames*, overheated extracts from journalistic articles on hackers, science fiction stories from the 1950s, contemporary reports on the failed videotext networks of the 1980s, and utterly dated predictions from the 1970s about the future of personal computing. None of these pieces of writing is assigned purely for its factual content (which is dubious), literary merit (often limited), or intellectual rigor (sadly lacking). Rather the point is to get students to examine them critically, and uncover the kinds of social influence and unexamined assumptions that we often cannot see in contemporary materials because we are too close to them. With liberal arts students at an elite private college, and a class size of around a dozen, the course was a reasonable success (at least the students gave good evaluations and enjoyed it, even if they didn't always do all the reading).

As other contributors to this volume demonstrate, one faces an enormous hurdle in trying to use techniques of this kind with computing students. Computing students tend to assume that courses will provide a large volume of factual technical information, which the majority would like to be of direct and obvious relevance to potential employment. They expect to cram this material into their head shortly before an examination, working from some kind of bullet-point list. They are not necessarily predisposed to respect the kind of "fuzzy" or "touchy-feely" activities valued by historians, such as group discussion, evaluation of arguments made by others, or the analysis of technical systems within a broader social context. If they had wanted to do those things, they probably wouldn't have picked the major they did. The particular problem is that computing students are disinclined to accept that study of particular examples now remote in time and space will teach them much of value.

The other obvious problem with computer science instructors trying to use history to help students think more like historians is that computer science instructors themselves do not know how historians think (any more than most historians understand how computer scientists think).

³⁶ Syllabi for my old courses are available from <http://www.tomandmaria.com/Tom/teaching>.

Overcoming this problem would require a profound and long-term realignment of the two disciplines.

History and the Core Curriculum

Yet the strategic potential is real, if hard to tap. Consider the current (2001) version of the joint ACM/IEEE-CS model curriculum for computer science education. History is conspicuous by its absence. Admittedly, the core curriculum does include history as a topic, but it allocates exactly one hour to it (of the sixteen devoted to “social and professional issues”), during which time students are expected to cover “prehistory—the world before 1946,” “history of computer hardware, software, networking” and “pioneers of computing.” That must make for a rather packed hour and, I fear, a confused and confusing one.³⁷

On the other hand, if we look at the five “characteristics a successful graduate should possess” at the end of a computer science program, provided elsewhere in the same report, we see enormous potential for history. Of these five key areas, at least three can be addressed with history.³⁸

Adaptability: If we can give students a sense of the unfolding of history and the many profound shifts in technology over the past few decades, they will be much less likely to assume that the current technology is the be-all and end-all. Less experienced practitioners often have a sense of only one generation of technology before the current one, and a foolish idea that if everything can just be migrated to the latest paradigm and really “done right,” then it will last forever. A better historical understanding means that students are: 1) less likely to be trapped as specialists in single technologies; 2) more likely to understand the need for flexibility and modularity in design; and 3) less likely to throw away old but serviceable systems and embark on pointless migrations.

System-level perspective: The same factors are at work here. An understanding of the evolution of any real system would expose students to the historical layering of technologies and interfaces. They would be better equipped to maintain code (something CS education does little to equip students for) and might be better able to develop a sense of WHY we must change some parts of a system and leave others alone, or why re-implementation is not always the answer.

Appreciation of the interplay between theory and practice: History teaches us that there are always tradeoffs between resource usage—including of course the resources of time, risk, maintainability, cost, and skill, as well as processor memory—and disk usage. Many of these tradeoffs have shifted profoundly with time as hardware costs have dropped spectacularly. Yet, as we all know, students may place an emphasis on optimizing code that makes no sense whatsoever, given the profusion of other resources and the need to lower cost and increase maintainability. How many database students learn when *not* to normalize? Some of what we teach students may reflect historical conditions that no longer hold. To generalize this a little, many techniques have niches within which they work well. The hardest thing is to teach students when each is appropriate, and when it should not be used. History can help us here.

³⁷ <http://www.computer.org/education/cc2001/final/sp.htm#SP-ProfessionalResponsibility>.

³⁸ <http://www.computer.org/education/cc2001/final/chapter11.htm>.

Conclusion

Let me finish, then, on this note of hope. Professionally trained historians are taking an increasing interest in the history of computing, and are working cooperatively with interested computer scientists and business and scientific computing pioneers. A diverse and growing literature covers the history of computing from many different perspectives, albeit far from comprehensively. There are real opportunities for the incorporation of history into the core computer science curriculum, in both tactical and strategic ways. More immediately, having knowledge of the history of your own field will make you a better teacher and a better researcher.

Acknowledgments: While this paper has the same basic structure as that presented at the 2001 Amherst workshop, “Using History to Improve Undergraduate Education in Computer Science,” I have been quite shameless in plundering good ideas and acute observations presented elsewhere during that workshop, and in the two smaller events held during 2004 to support the creation of this volume. The contributions of William Aspray, Martin Campbell-Kelly, Atsushi Akera, Michael Mahoney, and Thomas Drucker were particularly important in shaping my thinking.

Thomas Haigh is an assistant professor in the School of Information Studies at the University of Wisconsin, Milwaukee. Haigh holds a Ph.D. from the University of Pennsylvania, and has published articles on the history of the software industry, the origins of the database management system, the historical relationship of computing to corporate management, and the creation of the data processing department. His current research projects include the social history of the personal computer and the origins of open source software in the scientific computing field. He serves as Biographies Editor of *IEEE Annals of the History of Computing*, and is a member of the Committee on ACM History. *Author’s address:* SOIS, Bolton Hall, Room 568, P.O. Box 413, Milwaukee, WI 53201 USA. *Website:* www.tomandmaria.com/tom. *Email:* thaigh@acm.org.

II. Curricular Issues and Strategies

3. The Challenge of Introducing History into a Computer Science Curriculum

Paul E. Ceruzzi
National Air & Space Museum, Smithsonian Institution

Abstract

Introducing the history of computing into a computer science curriculum can benefit both students and their professors. The richness of historical material can provide benefits to beginning as well as advanced students. However, one must be clear at the outset what “history” really is, and what its contributions can be. Above all one must not do violence to history to fit the needs of computer science instruction. This essay looks at some examples where history was employed in the study of the classical sciences of chemistry and physics to seek guidance in adopting such a policy for computer science.

The study of the history of computing has much to offer to those who are studying or teaching computer science, computer engineering, or information technology at the college level. There are many possible uses, including:

- Historical material may assist faculty in computer science or related fields who wish to liven and flesh out their syllabi with solid factual material, rather than repeating stale, factually incorrect, or misleading anecdotes about “nerds” or other topics.
- For students at advanced levels, history can place the technical or scientific advances of computing into an economic, social, legal, and political context, giving them a realistic but not condescending sense of what they will soon experience upon graduation.
- History is an excellent vehicle to assist in the socialization of beginning students: it can show them what kinds of people have shaped the field over the years, their ethnic and economic backgrounds, what their values were, and what kinds of day-to-day work they did in their careers. History can let students know of the excitement, richness, and diversity of work in the field, and give them a sense of how they might someday be a part of it.
- History may be a way to introduce basic concepts of computing to students considering a major in these topics, with the assumption that concepts such as memory, processing, I/O, and so on are more easily grasped when looking at machines such as the Comptometer than a modern microprocessor.
- History may introduce the principles of computing to non-majors as well. A study of the history of computing is a much better way to provide the “computer literacy” necessary to function in a digitized world, in place of the traditional “how-to-surf-the-Web” courses, which become obsolete quickly. This may be tailored for beginning as well as advanced students (who have already chosen to major in another field). It can be especially useful for non-traditional students who are returning to school after years in the home or workplace, for whom a grasp of the social and technical milieu of computing can advance their careers and enrich their lives.
- Finally, introducing history into a computer science curriculum may enhance the quality of research done by historians. When historians become involved in this process, they will encounter topics and students who, thanks to Moore’s Law, will be part of history before long. The fact that such programs provide employment for historians is noted as well.

In order for any of these benefits to accrue one must introduce these topics carefully and with an understanding of both the methods of instruction and of the goals. These issues have been discussed before, by some of the most eminent 20th Century scholars, and it is worth looking at how they approached this topic for instruction in the classical sciences. What follows is a look at the historical context of such an effort.

James Bryant Conant and the Harvard Program

After World War II, James B. Conant, President of Harvard (Figure 1), developed what he called a Case Studies approach, using the history of science to teach undergraduates about science. It was based on a notion somewhat like the naturalist's rule that "ontogeny recapitulates phylogeny": that one can get a better understanding of modern science, in all its complexity, by tracing the evolution of scientific progress over time. Conant did not believe that this was a way to teach students the elements of a modern scientific discipline, but rather to convey a sense of the richness of the life of science to Harvard undergraduates, many of whom would assume positions of leadership in the United States and who presumably would benefit from having an understanding of what it was that scientists did.

Figure 1. James Bryant Conant (1893-1978)



(Photo: Courtesy of the Nuclear Age Peace Foundation
<http://www.nuclearfiles.org/rebios/conant.htm>)

A famous example from his program was to have students learn the chemistry of combustion based on the phlogiston theory as if it were "true," not (as we know now) "false." I put the words "true" and "false" in quotation marks because that is precisely the whole point of this approach; namely, how a scientist determines whether a theory is true or not. The philosopher Karl Popper described a valid scientific theory as one that can be tested, and that has the potential to be falsified by experiment. Popper's concept, which underlies modern science, is that there can be no direct proof of a theory's truth, only an absence of disproof. Conant's Harvard program gave his students a hands-on experience of Popper's fundamental definition.

Conant's approach did not find much popularity outside of Harvard. That did not mean, however, that his idea bore no fruit. Among the instructors at Harvard who assisted Conant in the preparation of course material was Thomas Kuhn, on whom this concept made a deep impression. To greatly telescope the subsequent events, Conant's approach was a major influence on Kuhn's own work, including and especially *The Structure of Scientific Revolutions*, one of the most influential books of the twentieth century.³⁹ Central to the thesis of that book is the notion, grounded in history, of the contingent nature of scientific truth with its argument—shocking at the time when he first expressed it—that new theories overtake older theories not because they are more correct, but because the adherents of the old theory die off (quoting a statement made by Max Planck about quantum physics). It was through Kuhn's book that history, as it is practiced by historians and not as it was conceived by scientists, was introduced into the study of science.

Note the influence of Kuhn's book. It has not been on the practice or the teaching of physics and chemistry. Few introductory physics or chemistry courses, to my knowledge, deliberately teach students the phlogiston theory of combustion. But *The Structure of Scientific Revolutions* has had an enormous influence on the social sciences: political science, psychology, sociology, the history of science, and so on. The word "paradigm," introduced by Kuhn to describe the framework under which most day-to-day science is conducted, has entered the daily vocabulary in a myriad of ways that Kuhn hardly could have imagined. His theory was acknowledged by geologists as having influenced those who championed the Plate Tectonic Theory, but that may be an exception that proves the rule. Kuhn later was dismayed by the many ways his concept of the "paradigm" was used, but we now see that his writings were among those that helped introduce the concept of "social construction" to the history of science and technology. In the years immediately following the publication of *The Structure of Scientific Revolutions*, critics focused on how Kuhn seemed to refute Popper, who had suggested that science proceeded in incremental, logical steps. Looking back on those writings from a twenty-first century perspective, they do not seem as far apart.

I would not argue that undergraduates beginning a course of study in computer science be familiar with the intricacies of the social constructionists, or the debates over Kuhn's vs. Popper's definition of science. But a basic understanding of those concepts ought to be part of any curriculum that introduces students to the scientific method. Historians certainly know the "paradigm shifts" (with apologies to Kuhn) that transformed computing from the punched-card, batch era, to time-shared mainframes, to personal computers, to a client-server architecture and a packet-switched global network. If computer scientists regard this progression as logical and incremental, they are wrong and students ought to be taught otherwise.

Stephen G. Brush

There is a second instance where a scholar looked closely at the teaching of the history of science and its relationship to science. Thirty years ago, Stephen G. Brush, of the University of Maryland, published a famous paper with the provocative title "Should the History of Science be Rated X?"⁴⁰ Again I am telescoping the arguments in that paper, but one thing that Brush noted was the discrepancy between science education, with its grounding in Popper's logical positivistic foundations, and the reality of real scientists, who the historians tell us have plagiarized, fudged data, deliberately ignored results they did not like, resorted to *ad hominem*

³⁹ Thomas S. Kuhn, *The Structure of Scientific Revolutions*, Second edition, enlarged (Chicago, 1962).

⁴⁰ *Science*, 183 (22 March 1974), 1164-1172.

attacks on anyone who disagreed with them, and so on—everything that a scientist is not supposed to do.

For those who would use history as an aid to the socialization of incoming freshmen intending to major in computing-related topics, this essay might give one pause. Even a less extreme reading of Brush's argument leads to a conclusion that history and science have little to offer each other. Referring to Martin Klein, he states, ". . . history and science are inherently different kinds of disciplines; bringing them together is likely to do violence to one or the other."⁴¹ Note that Brush was writing at a time when scientists were concerned about their public image as "robot-like" and "lacking emotions and moral values."⁴² For Brush, the introduction of history was a way to show students that science was fundamentally a social activity, albeit one in which logic trumped emotions or politics.

As applied to the history of computing, Brush's quotation of Klein's critique may not be a serious objection. Historians have been able to draw on several decades of experience, debates, and discussions—especially in venues such as the meetings and journal of the Society for the History of Technology (SHOT)—concerning the differing values of historians and scientists and how they need not be antagonistic to one another. Papers from that society's journal, *Technology and Culture*, show that one may invoke the theory of social construction without resorting to the extreme view that the basic laws of physics are of only marginal relevance to modern technological systems. The writings of Thomas Parke Hughes, for example, acknowledge the genius and dedication of computing pioneers such as Jay Forrester or Elmer Sperry without making them two-dimensional "robots" who had no existence outside pure logic.⁴³

Still, there are differences between the practice of history and of science and it is worthwhile keeping them in mind. Brush's concern about the public image of the scientist may no longer be relevant today, when computer scientists are publicly valued for their ability to generate economic wealth, provide high-paying jobs, and allow the United States to remain competitive in world markets. Meanwhile, in television programs, movies, and "techno-thriller" novels, computer scientists are the new superheroes, saving the world with a few deft strokes of a keyboard or hacking into an enemy's system and disabling it at the last possible second. (Note that Hollywood filmmakers often cast attractive young females in these roles, whereas in Brush's day they were almost always men, like James Bond, who were a few years older.) The study of history can be a useful way to provide students with a realistic idea of the work they will be expected to do, just as Brush hoped in his day that the study of history would counter the false impressions of scientists that were prevalent in the 1970s. This does not rule out the possibility that some of today's computer science students will have a career that James Bond would envy, but one would hope that the students will appreciate that the intellectual rewards of doing good computer work are more than sufficient.

The Nature of Modern Computer Science

Lastly, if one is interested in the value of history for the teaching of computer science and related disciplines, it is worth considering the nature of "computer science" itself. The term has an interesting history, and it emerged after some debate over alternatives (for example,

⁴¹ Ibid., 1166.

⁴² Ibid., 1171.

⁴³ See, for example, Thomas Parke Hughes, *American Genesis: A Century of Invention and Technological Enthusiasm* (New York, 1989).

“Informatics,” which caught on in Continental Europe but not so much in the Anglophone countries).⁴⁴ That debate, whether “computer science” is really a science or not, is relevant to this paper because it indicates that some of the arguments of Conant and Brush, whose focus was on the classical sciences of chemistry and physics, may not apply here.

The arguments of the 1960s about whether computer science was a science or not revolved around the question of whether the stored program digital computer, as an object of study, was of the same class as the natural phenomena that chemists and physicists studied. Allen Newell and his colleagues argued that it was, even if the computer was an artificial, not a natural, entity. But there are other reasons to argue that computer science is not a science like chemistry or physics. Computer science has about it many of the trappings of what I would call a “postmodern” science. The characteristics of such a science are rewards and stature for its practitioners that are less a matter of peer review and publication of seminal papers than of publicity, fame, and, above all, commercial success—especially in Silicon Valley. Professor Reinhard Keil-Slawik, of the University of Paderborn, has lamented that when computer scientists publish research papers they seldom cite other papers that are more than a few years old; they lack the broad understanding of a need to “survey the literature” that characterizes the work of many classical and quantum physicists.⁴⁵ Computer science moves too fast for that. And this is not a recent phenomenon: beginning in the vacuum tube era, Alan Perlis of Yale University remarked that “Computer science is embarrassed by the computer.”⁴⁶ The leaders of post-modern sciences typically do not simply hold faculty appointments at major universities; rather, they are directors of “Institutes” or “Laboratories” that are affiliated with a nearby university (and may even be on campus), but are otherwise autonomous and have enormous budgets that are independently funded and administered.

Examples of such practitioners from computer science might include Steven Wolfram, known for the financial success of his program *Mathematica* and for his interviews on National Public Radio and on the Charlie Rose television show to discuss his latest book, *A New Kind of Science* (note the title). Or Ray Kurzweil, known for his provocative books, op-ed pieces, and personal friendship with the musician Stevie Wonder.⁴⁷ Or Nicholas Negroponte of the MIT Media Lab, a place he founded with the goal of “inventing the future,” even though many of the things that make up our current multimedia—such as the World Wide Web, browsers, search engines, file-swapping music programs, and so on—came from elsewhere.⁴⁸ In May 2004, MIT’s Computer Science and Artificial Intelligence Lab moved from its humdrum facilities at Technology Square to the new Ray and Maria Stata Center for Computer, Information and Intelligence Sciences, designed by the famous “postmodern” architect Frank Gehry and reported to cost about \$285 million.⁴⁹ A description of the building quoted William Mitchell, MIT’s Dean of Architecture, who said, “To be successful today, a lab needs a strong brand, especially in dealing with the corporate world.”⁵⁰ Clearly the researchers who are moving into that building will be expected to pay their own way, somehow.

⁴⁴ Allen Newell, Alan J. Perlis, and Herbert A. Simon, “Computer Science,” Letter to the editor, *Science* 157 (22 September 1967): 1373-74.

⁴⁵ Keil-Slawik, paper presented at the History of Software Engineering Conference, Dagstuhl, Germany, 30 August 1996.

⁴⁶ Alan J. Perlis, “Epigrams on Programming,” *ACM SIGPLAN Notices* (October, 1981), 7-13; Epigram #112.

⁴⁷ Kurzweil, *The Age of Spiritual Machines: When Computers Exceed Human Intelligence* (New York, 1999).

⁴⁸ Stewart Brand, *The Media Lab: Inventing the Future at MIT* (New York, 1987).

⁴⁹ Spencer Reiss, “Frank Gehry’s Geek Palace,” *Wired* (May 2004): 194-99, 211-12.

⁵⁰ *Ibid.*, 212.

Among computer scientists today, perhaps Donald Knuth of Stanford fits the classical model of a research scientist who is not asked to justify his work in terms of the money it brings to his host university. No doubt there are many others like Knuth who are less well known. But the connection between computer science and the technological wizardry that emanates from Route 128 or from Silicon Valley makes it difficult to conceive of the study of computers as a classical science. Undergraduates already know this, albeit imperfectly, when they choose to major in computer science or related disciplines.

Conclusion

That leads to a paradoxical conclusion that, while the study of history has little to offer to those who are teaching young people to become scientists in the classical sense, the study of history can have great value in teaching young people to become practitioners of post-modern science, of which computer science is an example. Knowing the messy and tangled paths to the successful practice of computing requires just the kind of sophistication that, years ago, James Bryant Conant sought for his Harvard undergraduates, who were being trained not to become scientists but politicians, diplomats, financiers, and businessmen. Keil-Slavik need not worry, unless one feels that computer science should not be that way. But if all sciences are trending in this direction, then the study of the History of Computer Science is very important indeed.

Paul E. Ceruzzi is Curator of Aerospace Electronics and Computing at the Smithsonian Institution's National Air and Space Museum in Washington, DC. He received his PhD in American Studies in 1981, and was the second recipient of a Tomash Fellowship from the Charles Babbage Institute, which supported his graduate work. He is the author of several books on the history of computing and related topics, the most recent being *A History of Modern Computing*, Second Edition, published by MIT Press in 2003. He is currently working on a book-length study of Systems Integration firms located in the Washington, DC region. *Author's address:* National Air & Space Museum, MRC 311, P.O. Box 37012, Smithsonian Institution, Washington, DC 20013-7012 USA. Email: Paul.ceruzzi@nasm.si.edu.

4. History in the Computer Science Curriculum

J.A.N. Lee
Virginia Tech

Abstract

Understanding the history of computing is an essential element of the curriculum of technology, and no less so within the curriculum of computer science and information technology. While being a recent history, the number of generations is similar to those of a technology with a much longer record. While many faculty have an interest in the subject, incorporating computer history into the curriculum suffers from the lack of readily available resources that can be easily assimilated into lesson plans. Consequently, using history to improve the undergraduate computer science curriculum is being thwarted by the lack of resources. This paper explores the opportunities for improvement in resolving this shortcoming.

Looking at the process and practices of teaching in almost any subject, teaching depends heavily on history, and, in many aspects, all teaching is the teaching of history. While educational methodologies may vary widely, the resources used as the basis for undergraduate learning are archived in the history of the field, and only in those exceptional courses, such as in an honors program, is there development or enlargement of knowledge through the educational process. The difference between teaching history and using history is primarily in how far back one looks at the past. Courses in modern technology, such as computer science, tend to rely heavily on the current state of the art to ensure that students graduate with a background that will enable them to integrate into the workforce with ease. This shallow view of the field deprives learners of an understanding of the life cycle of the subject and overlooks the development phases that led to the current state of the art. In a time when most undergraduates in today's universities and colleges have no knowledge of a world without the personal computer, there is a need to give them a sufficient view of the stages of growth of the subject to ensure that they understand where it came from and the rationale for the current state. Incorporating the history into the curriculum also allows the student to realize that there are human elements to the field whose contributions should be recognized and lauded.

It is appropriate that a clear distinction be made between a course in the history of a subject and incorporating history into the curriculum, and between teaching history as part of the curriculum and requiring students to take a course in history. Many "History of Technology" courses now contain a module on the development of the computer, though some feel that contemporary history is still not a subject of study for true historians. In the same vein, computer historians are not highly sought by either departments of history or computer science.

In examining the resources available to assist in the teaching of the history of computing, we will see that while there is a growing collection of materials to support a full course in the topic, there are few resources to assist the average teacher to incorporate history into another course such as "operating systems" or "computer architecture." We hope to overcome this through the continuation of this project.

Incorporating history into the CS curriculum has strong parallels with the ongoing work to incorporate ethics and social impact into the curriculum. With the impetus provided by the Computer Science Accreditation Board in 1984, computer science programs throughout the United States began to find ways to slip ethics and social courses into the curriculum to meet

accreditation requirements. However, it has taken almost fifteen years to reach a stage where resources are becoming available to support not only separate courses in ethics and social impact, but also to provide teaching modules for other courses. This workshop report is the starting point for replicating that activity for the history of computing.

Curriculum 2001⁵¹ was expected to include a module on the history of each subject when completed, but the Steelman version only prescribes a study of the historical elements in relation to Programming Languages, Net-Centric Computing, Intelligent Systems, and Information Management. The “Social and Professional Issues” module includes a general overview of the history of computing. Perhaps this current publication will provide another impetus for the identification of appropriate resources to assist faculty in preparing to teach courses that conform to the recommendations.

Within the CS curriculum, the inclusion of a study of historical elements has the advantage of requiring CS students to “think outside the box” for a while, and particularly outside the box that is defined by the PC sitting in front of them. Provided that a study module is more than a lecture-style presentation and that students are expected to “research” the topic themselves, it will force CS students to look beyond the machine and language manuals, and thereby broaden their outlook on the world. History modules must be more than just the recitation of anecdotes, though this may be a good place to tweak their interest. Using history as a subject of study will have the advantage of requiring students to do some library research beyond the World Wide Web, to write more than comments in program headers, and to perform some critical thinking that is not necessarily algorithmic—all subjects that are not the favorites of CS majors, but which need reinforcement. Computer science is not famous for its use of case studies as an educational methodology, but history may be able to provide the resources to foster this approach to CS education.

The major purpose of inserting history of computing into the undergraduate CS curriculum is to improve undergraduate education, not simply because it is a good thing to do. While there are the advantages suggested above, there is also the need to provide vehicles to improve the learning of standard topics. If it takes anecdotes about the pioneers of the field to provide an instrument by which a student remembers the concepts of a topic, then that is a small success. If the study of a topic includes the progression of innovation (hopefully also examining the failures along the way) leading to the justification for the characteristics of current artifacts, that is an improvement.

Opportunities

There are three ways to insert the history of computing into the curriculum: 1) by installing a separate course with the content and title, “The History of Computing”; 2) by including a module on its history into each subject, as suggested in “Curriculum 2001”; or 3) by using history as a teaching methodology, truly integrated with the subject. Let us examine each of these options.

The opportunity to install a completely separate course may be difficult in many ways. First, the course is not required by current accreditation criteria, and second, few computer science programs have a faculty member who is prepared to teach the course. Furthermore, “diluting” the CS curriculum by adding a new three-credit requirement may mean that some

⁵¹ ACM/IEEE Computer Society, “Computing Curriculum 2001,” (Steelman Report), <http://www.computer.org/education/cc2001/steelman/cc2001/index.htm>.

other topic has to be dropped. In those instances where the total number of credit hours is limited by institutional restrictions, such as limits on the total number of hours in the major, there may be an advantage to cooperating with the history department to offer a special course to CS majors that would count as part of their "humanities" core requirements instead of the major requirements. This does require that the instructor be as knowledgeable about computers as the participants, and has sufficient technical background to appreciate the nuances of development cycles. At this time, there are few computer scientists who have had additional training as historians, and few historians who have the computer science background or training in the history of computing.

To teach a "full" course also requires that there be a primary textbook and access to other readings. A few books have the potential for fulfilling this requisite, but the recent publication of a syllabus by the International Federation for Information Processing (IFIP) should provide the motivation for others to be added to this short list.⁵² The syllabus was the result of the work of an international panel of computer historians from IFIP Technical Committee 3 (Education) and Working Group 9.7 (History of Computing). The resulting document was approved by IFIP and was published in the *IEEE Annals of the History of Computing*.⁵³ While this document is organized to provide individual modules related to the seminal topics with computer science that together constitute a full course, the integration of these modules with other technical subjects was not the primary thrust of the work.

As has been found in creating a course in ethics and social impact that is separate from the rest of the curriculum, a separate course in the history of computing has the implication, in the minds of many students and even some faculty, that it is distinct from the rest of the curriculum and is perhaps not as important as the technical subjects. Is it a contribution to the resume of the average graduate?

The second option is to install an historical module in a variety of courses so that the sum total is roughly equivalent to the three-credit course. This implies that rather than having a single computer historian associated with the CS program, it is now necessary for all faculty members to have a "smattering" of computer history, sufficient to cover their syllabus obligations. Alternatively, the module for each course may be taught initially by the local computer historian, and thereafter taken up by the regular instructor. As long as the schedule can be organized so that not every course starts the term with the history module, this "on-the-job" training can be successful.

Another alternative may be to organize a department-wide "While you are away" plan to cover for faculty when they are away from classes to attend conferences and other meetings. Here their substitute is the computer historian who will use that opportunity for a presentation (and hopefully an assignment) on the history of the subject. This ad hoc approach, however, does not ensure that the history module is covered on a regular basis or is truly integrated into the syllabus. Sometimes class periods are so precious that traveling faculty members would prefer to have a topic of their own choice covered in their absence. Several textbooks pertinent to the early courses in computer science have chosen to include a section on the history of computing. This is laudable, but lacks two important properties—integration with the subject and correctness. These are textbooks primarily written by authors who have had to rely on

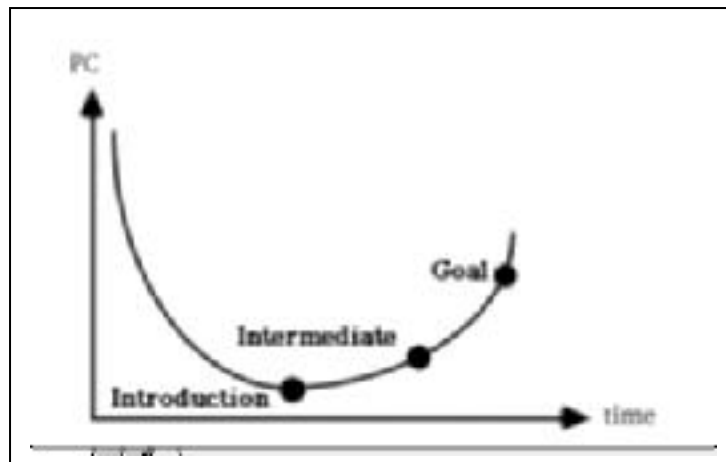
⁵² IFIP, "History in the Computing Curriculum," Joint Task Force WG9.7/TC3 (1998), http://www.hofstra.edu/pdf/CompHist_9810rprt.PDF.

⁵³ John Impagliazzo, Martin Campbell-Kelly, Gordon Davies, J.A.N. Lee, and Michael Williams, "History in the Computing Curriculum," *IEEE Annals of the History of Computing* 21/1 (1999): 4-16.

secondary sources for information, and thereby merely serve to further propagate myths. The most common include describing the architecture of personal computers as the “von Neumann architecture” without having read the “Draft Report on EDVAC,” and attributing the development of COBOL to Grace Murray Hopper.⁵⁴

In the same way that many faculty have incorporated technology into their courses—through the use of overheads, slides, the World Wide Web, and presentation graphics—have chosen to approach the task of teaching through educational technologies such as the Keller Plan, active learning strategies, and others, incorporating history can be a teaching/learning enhancement. By implication, the sequence of introducing a subject may be the historical sequence of development. This has the advantage that the subject can be presented from the simplest to the more complex, showing the innovations that provided stages of development rather than jumping into the morass of current complexity. Let us consider one such example of using history as the template for a lesson plan. In Figure 1 below, the “Perceived Complexity (PC)” of a concept is measured in relation to the time from inception to current application.⁵⁵ At the initial introduction, the PC is high, simply because the idea is new and, perhaps, not well explained. As understanding is attained, the PC drops until the measure reduces to a minimum when full acceptance is achieved. Once accepted, the purveyors of the product then make improvements and add enhancements, increasing the PC continually. This curve (sometimes called the “paradox of technology”⁵⁶) can be recognized in a large number of computer concepts, hardware or software developments.

Figure 1. Perceived Complexity



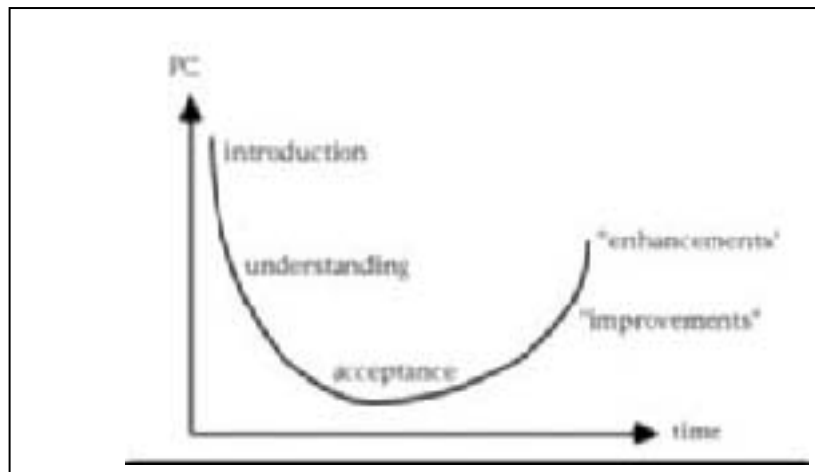
A complete history of a subject would cover the whole PC curve, but a better approach to teaching the subject would start at the point of minimum PC and then progress through the remainder of the curve as the subject is presented, as shown in Figure 2.

⁵⁴ M.D. Godfrey and D.F. Hendry, “The Computer As von Neumann Planned It,” *IEEE Annals of the History of Computing* 15/1 (1993): 11-21.

⁵⁵ J.A.N. Lee, “Perceived Complexity: A Conjecture Regarding the Life Cycle of Computer-Related Products,” *IFIP World Computer Congress 2000*, Beijing, China (August 2000).

⁵⁶ Donald A. Norman, *The Psychology of Everyday Things* (New York, 1988).

Figure 2. Teaching Along the PC Curve



That is, the introduction to the subject starts with a description of its “state” at the point of lowest PC, and then historical complexities are added to that as learning progresses. History will provide the elements of this progression and the successive stages of presentation. Even if the complete curve is not utilized, the PC curve does provide an indication of the “most understandable” or “more simplistic” concept with the domain of discourse that can be used as the introduction to the topic.

Concerns

We have a syllabus for a full course that also includes some suggestions for creating learning modules in a limited number of topics, though these tend to be more related to periods in history than to specific subjects. Teaching history as a pedagogical tool does not mean that this version of history is simply the teaching of dates and related facts; it is important that this historical approach be related to the technical subjects that are the real topic of study.

Looking at it another way, we need to develop orthogonal views of computer history that cut across the timelines and emphasize the history of specific topics such as operating systems or human computer interaction. The development of these views probably cannot simply be left to historians to construct, but will necessarily involve practitioners as well as pioneers. While the current work towards a National SMETE Digital Library (<http://smete.org>) through the CITIDEL (Computer and Information Technology Interactive Digital Education Library) project (<http://www.citidel.org>) plans to provide access to online computer history materials, the assimilation of resources into lesson plans is a much more difficult problem. The lack of assimilated resources is perhaps the reason behind the lack of specification of studies of computer history in many of the learning modules in Curriculum 2001.⁵⁷

It is probably unlikely that commercial publishers will provide textbooks in the field that cover specialized studies of computer history in individual subjects. However, we should be encouraging authors of technical textbooks to include a section on the background history,

⁵⁷ ACM/IEEE, “Computing Curriculum 2001,” op. cit.

authored in cooperation with an historian. Similarly we need to encourage online virtual museums to go beyond being collections of images to providing “walk-through” galleries that will provide deeper descriptions of specialized histories. In fact, museums should merely be portals to distributed digital libraries-to resources with intelligent assimilation systems to permit teachers and students to construct custom-selected study guides. One such resource could be constructed from the volumes of the *Annals of the History of Computing*, which already contain (but not in a digitized form as yet) many of the stories of computing distributed across formal articles, memoirs, anecdotes, and obituaries.

Clearly there is a need to persuade granting agencies and foundations to support the development of these resources. There has been a reluctance to fund extensive projects in the history of computing with the excuse that the principals are not “trained historians,” in a time when the number of so-qualified individuals is extremely small. Just as many early computer scientists themselves were not educated within the computer science discipline, computer historians are primarily “self-grown.” For many years the impetus for recording and recovering the history of computing must be led by computer scientists with advice from historians of technology in order to create the encyclopedia of the history of computing to support computer education.

John A.N. (JAN) Lee is professor emeritus of computer science at Virginia Tech, having spent 47 years in teaching and research, starting in civil engineering in the 1950s and converting full time to computer science in 1964. As a civil engineer he was involved in the design of several structures around the world that are well known, including the Sydney Opera House and the Firth of Forth Road Bridge. Before getting deeply involved in the history of computing he was involved in national and international standards, leading the work on the standards for PL/I, BASIC and Ada. In the last twenty years of teaching he was a champion for the introduction of history, ethics, and social awareness into the computer science curriculum. He is a Fellow of the Association for Computing Machinery and of the Institute for Distributed and Distance Learning. He is the author of eight books and more than 400 articles. *Author's address:* 4247 Horseshoe Loop, Dublin VA 24084-5863 USA. Email: janlee17@verizon.net.

5. Using History in a Social Informatics Curriculum

William Aspray
Indiana University in Bloomington

Abstract

This paper describes three courses taught in the area of social informatics at the School of Informatics at Indiana University in 2003 and 2004: Organizational Informatics; Internet and Society; and Privacy, Security, and Information Warfare. The paper describes the role of social sciences in broadening the education of Informatics students. Discussion includes the use of history in each of these courses, and the limited value of history compared with the other social sciences in a presentist-oriented curriculum. The complete syllabi have also been compiled separately in Part III of this volume.

This paper describes the use of history in a social informatics curriculum from my experiences over the past two years teaching undergraduate and graduate classes in social informatics in the recently formed School of Informatics at Indiana University.

Some background is needed to understand the purpose of informatics and of the social informatics courses within Informatics. Over the past six years, there has been a significant movement in the United States to form new kinds of computing educational programs. Some forty to fifty programs have been formed in the United States and Canada. These are sometimes called “information schools” or, in the case of Indiana University and a few other places, “informatics” schools or programs. The term “Informatic s” is not used in the same sense as it is in Europe where it denotes programs in what Americans would call “computer science.” Instead, in North America it denotes an interdisciplinary program focusing on the study of computing in application and context. (Some schools might replace the word “computing” in the previous sentence by “information technology” or possibly by “i nformation technology and information”.)

Informatics was in part a response to the high demand for information technology workers during the dot-com boom of the late 1990s and 2000. Thus, in a number of universities, the informatics/information school was intended primarily as an educational program to provide an alternative to the narrow technical focus of a traditional computer science education. Its goal was to attract students who had an interest in computers and the Internet but perhaps less interest in hard-core computer science, while meeting employers’ demands for students with more breadth in their education, including “soft” skills such as teamwork and communication skills.

In another set of schools, however, informatics was seen as a new kind of interdisciplinary research discipline. These research-oriented programs fall basically into three types. There were library and information science schools (e.g., Michigan, Washington, Syracuse, UC Berkeley, Texas) that broadened their subject coverage and faculty expertise to include more technology and cover such topics as information retrieval, human-computer interaction, and social studies of information technology in action. There were computer science departments (e.g., Carnegie Mellon, Georgia Tech, UC Irvine) that broadened their traditional computer science coverage to include faculty members who were social scientists, cognitive scientists, designers, and artists to study human-computer interaction, usability, organizational context, and security. Lastly, there were completely new ventures (Indiana School of

Informatics, Penn State School of Information Science and Technology) formed to conduct this interdisciplinary study, often side-by-side with a computer science, or even a library and information science, program.

Indiana University School of Informatics

Indiana's School of Informatics studies bioinformatics, chemical informatics, and other areas of science informatics; human-computer interaction, social informatics, new media, music informatics, and other human-centered informatics disciplines; web, text, and data mining, which is known in the School as "Discovery and Application of Information"; cybersecurity and information privacy; mathematical and logical foundations of informatics; and modeling, simulation, and complex systems (including cognitive systems). The school currently offers bachelors and masters degrees, and approval for a doctoral program is expected shortly.

Social informatics is the use of social science approaches to study information and information technology. Methods from a wide variety of disciplines can come into play: history of science and technology, philosophy, sociology, anthropology, political science, science and technology studies, cultural studies, critical theories, communication and media studies, and other approaches. Social informatics has been taught in many different settings by individual faculty members (though not typically under that name), especially in engineering schools. It was also taught in some of the library and information science schools (notably Syracuse) and in the computer science departments (notably UC Irvine) even before they evolved into these informatics/information schools. Social informatics serves the same purpose as history in broadening education in the computing domain—the difference being that social informatics has a larger palette that includes not only history, but all the other social science disciplines. This paper discusses both, using social sciences to improve computing education and the specific use of history within these social informatics courses.

The undergraduate major in informatics requires students to take a course in social informatics, which, according to the School's online course listings:

"introduces the social and behavioral foundations of informatics. Theoretical approaches to how technology is used from psychological and sociotechnical perspectives. Examples of how current and emerging technologies such as games, e-mail, and electronic commerce are affecting daily lives, social relations, work, and leisure time."

This course is intended for sophomores. It is often followed by a junior-level course in organizational informatics. Although the organizational informatics course is an elective, it is one that fulfills programmatic requirements for most of our undergraduate majors and, thus, is taken by the majority of them. It is offered each semester, and almost always reaches its enrollment limit of seventy-five students. According to the School's online course listings, organizational informatics:

"examines the various needs, uses, and consequences of information in organizational contexts. Topics include organizational types and characteristics, functional areas and business processes, information-based products and services, the use of and redefining role of information technology, the changing character of work life and organizational practices, sociotechnical structures, and the rise and transformation of information-based industries."

In the first several years of the School's history, with a limited number of faculty in place, students took courses as they became available. The intended progression of courses was often not followed. As a result, many of the students who took organizational informatics did not have the advantage of having already taken the social informatics course and learned about the social study approach to studying technology or, generally, about the notion of socio-technical systems. The hiring of the social informatics faculty was still in process; thus, there had not been a systematic effort by the people who were teaching these courses to figure out what knowledge and skills sets should be learned in each of these courses and how the organizational course should build upon the social course.

The School also offers more advanced courses in social informatics. These include both an introductory graduate course on the Social Impacts of Technology and graduate-student-only elective courses (such as ones I taught on underrepresentation of women and minorities in the computing disciplines, and on the geographical advantages of high-tech regions). They also include elective special-topic seminars open to both upper-level undergraduate majors and graduate students. Two of these courses that I have taught—on Internet and Society and on Privacy, Security, and Information Warfare—are described below.

The remainder of this paper will describe my experiences in teaching three social informatics courses: Organizational Informatics (a junior-level course that I taught in fall 2002 and spring 2003), Internet and Society (undergraduate/graduate elective seminar that I taught in fall 2003), and Privacy, Security, and Information Warfare (undergraduate/graduate elective seminar that I taught in spring 2004). The syllabi for these courses are found in appendices to this paper.

Organizational Informatics

The organizational informatics course had an enrollment of seventy-five to eighty students each time and was taught in two lectures and a recitation session each week. One of the problems our social informatics faculty has with all of its large undergraduate courses is that there is no graduate degree program in social informatics; thus there is an uncertain and limited pool of graduate students qualified to be teaching assistants. I was fortunate to have been assigned a chemical informatics graduate student who had training in library and information science, an interest in social informatics issues, and a gift for teaching—even if no advanced training in social informatics. My colleagues have not always been so fortunate with the TAs assigned to them.

This was a fairly new course (having been taught only twice before by visiting instructors). The curriculum was far from set, and the students did not know what to expect. I found, as the term went on, that the students were hoping to continue to learn about technology rather than study the context, social shaping, and impact of technology. They continued, throughout the term, to request assignments in which they could use their technical skills—for example, to learn about or apply various kinds of business information systems that had been discussed in class or in the textbook.

I chose three goals for the course: 1) to bring in outside speakers who could give the students a sense of what it was like to work with information systems in various real-world settings; 2) to learn some of the basic concepts and facts about business information systems, since so many of the students would end up working in such an environment after graduation; and 3) to give them examples of analysis from various social science disciplines so they would

get an introduction to how social informatics could give them a perspective on the information technology with which they lived and worked.

Although there was significant variation in the quality of the outside speakers and the students' reception of them, this was the part of the course that was best received by the students. Many had very limited work and intern experience using information technology, and they were eager to learn what IT work life is like. Some students eagerly approached the speakers after each class, trying to learn whether that speaker's particular work environment would be right for them (and possibly trying to angle a summer internship!). Students related best to the younger speakers who were more familiar with student interests and they were better able to see the speakers as "possible selves." The first time I taught the course I had CEOs and CTOs speak, but I found the students were more interested in what programmers, systems analysts, and first-line managers did in their daily work lives.

I had been told by my colleagues at UC Irvine and Michigan that the students needed the crutch of a textbook for an organizational informatics course. (Irvine had taught such a course for about a decade when I started.) I chose the same textbook that was being used in several of these informatics/information schools, written by Steven Alter. It had the two virtues of being very good at standard textbook matters (clear writing, good layout, self-exam questions, etc.); and being organized around the concept of work, which students found intuitive and social informatics faculty could use as a basis for presenting their critical analyses. In hindsight, I believe it was wise to use a textbook and this one in particular, although many of the students pronounced the book to be boring.

The trickiest part of the course was to introduce the students to the approaches of the various social sciences. The university archivist came and gave several lectures on the nature and importance of metadata to electronic record-keeping systems, as an introduction to the approaches of information science. I presented the productivity paradox to the students in several lectures to introduce them to economic analysis. Wanda Orlikowsky's work on the reception of Lotus Notes in two organizations was used to introduce students to sociological and organizational management issues. The IT workforce shortage and workers on temporary visas were used to introduce the students to policy issues.

Lastly, I decided that the students should know something about the heritage of their field of study, so I provided six lectures on the history and structure of the information industries. I presented synthetic business history lectures on the predecessor industries (such as business machines), mainframe computer industry (two lectures), software and services industries, mini- and micro-computer industries, and the Internet industry. A small number of the students were deeply interested in this material. Most of the students, however, had trouble relating to the first three lectures because they had essentially no experience with mainframe computers and did not appreciate the relevance to their own lives. There was much stronger interest in the lectures on the microcomputer and Internet industries, since the students were familiar with both the technology and the companies involved. It would not be easy for someone who is not familiar with this material to prepare such lectures. The material is spread out over many different sources, and there are not many synthetic accounts. If I teach this material again, I will shorten the sections on early history and expand the materials on the period since 1990, despite the fact that very little historical literature exists on which to base these lectures.

Internet and Society

The second course was a seminar on Internet and Society, which had eight graduate and three undergraduate students enrolled. The course description is as follows:

"This course will cover the origins, governance, economics, operation, and use of the Internet. Governance and economic issues include who makes decisions about domain name and spectrum allocation, technical standards and innovation (IPV6, innovation in an end-to-end architectural environment, open and closed standards), and how the Internet is paid for. Operational issues include spam, worms, viruses, distributed denial of service, accommodation for those individuals with various impairments, and accessibility to a wide range of individuals (Digital Divide issues). Use issues include cybersquatting and trademark infringement, peer-to-peer music- and video-sharing, copyright and copyleft, use of the Internet for vices such as gambling and pornography, e-commerce and taxation, use by the government for e-voting and improved government services, and privacy considerations. Although some technical issues will be discussed, the focus will primarily be on social, political, legal, economic, and historical examination."

Internet and Society courses can take many approaches. (See the syllabus in Appendix A at the end of this chapter.) This course focused primarily on legal and policy issues, and to a lesser extent on economic and social issues. Some of the students were somewhat surprised not to find more discussion of topics such as online communities and computer-mediated communication such as blogs. But I believe that, without exception, every student became extremely interested in the material over the course of the semester.

This was a reading and discussion course, meeting once a week for three hours. The typical week's reading was an entire book or an equivalent number of articles. I have found that students pay more attention to a book if they are reading it for their own purposes rather than those of the author or the instructor. To encourage the students to read for some personally defined purpose, they were required each week to write up a question for discussion in class on some aspect of that week's reading that was interesting to them, but must also stand the test of being interesting to the instructor and the other students. Students could write up to one single-spaced page to give them the space to set up their question. I found that the students were diligent in looking for a question that the rest of the group would find interesting, and in doing so they became active readers. I was pleased by this outcome because many of the students, both undergraduates and graduates, were not accustomed to either heavy reading or reading a text critically.

The weekly submissions from students also served as a basis for the class discussion. I found that the submissions typically grouped into three to five major themes. Time would always be found to discuss these themes, plus other topics that I wanted to cover (for example, a critical analysis of the week's reading, setting the reading in context, or covering material about the week's topic that was not covered in the assigned readings). The level of the discussions was surprisingly good, given that many of the students (most of them human-computer interaction graduate students) had had few courses in their past education in the social sciences or even courses in which they had to read and critically discuss the material.

I supplemented the reading and discussion in three ways. First, students gave fifteen-minute presentations (they are hooked on Powerpoint!) on topics I assigned. For example, in discussing jurisdiction on the Internet, a student presented on the abandoned oil derrick in the North Sea that was turned into a sovereign principality to provide Internet and data storage services with customer privacy and protection exceeding that available in most other sovereign

nations. Second, I would frequently give lectures lasting from thirty to sixty minutes on topics related to the week's reading, such as e-voting or cybersquatting. Lastly, many weeks we showed videotaped lectures on Internet law from experts at leading law schools in the United States. In the future, I would select portions instead of showing the entire video, leaving more time for discussion.

Near the beginning of the semester we read Janet Abbate's history of the Internet. The students had moderate interest in the history itself, but the book proved useful later in the semester in helping them understand the unexpected ways that users redefined a technology (designed for specific military purposes). We also read Milton Mueller's book on Internet governance, which included some history of the various efforts by DARPA, ICANN, and others to govern the Internet, as well as the tensions between commercial interests, US and foreign government interests, technical communities, and user communities as they each attempted to exert control over this governance process.

The range of topics can be seen in the syllabus (see Appendix B). It is worth pointing out one, somewhat controversial, topic. Gambling and pornography are interesting topics when studying the Internet because they are among the most profitable of Internet-based businesses, the purveyors (especially of child pornography) are among the most technically skilled in order to evade law enforcement, and the wide reach of the Internet to supply these applications often comes into conflict with local community standards. We read an excellent ethnography of Internet-based child pornography by Philip Jenkins. The majority of the students found it enlightening to learn what the book has to say about issues such as pornography as an Internet business and technical evasion of law enforcement. However, several of the students had difficulty detaching themselves emotionally from the topic. They were upset and could not understand why I would assign such a topic.

Privacy, Security, and Information Warfare

The third and last course I will discuss here is a seminar on Privacy, Security, and Information Warfare open to juniors, seniors, and graduate students in Informatics. Compared with the Internet and Society course, the enrollment was twice as large and overwhelmingly undergraduate (twenty-one undergraduates and two graduate students). The class size would have been much larger if we had not set enrollment limits. The course was intended to meet a strong student demand for a security course, but also to educate me so that I could help the School make better informed decisions about hiring and curriculum development in this area. The course was also considered a prototype for a course on the social perspective on security that would complement a planned suite of technical courses in this area.

I had some trepidation about teaching the course because of my limited familiarity with the subject (no research in the area; limited teaching of the topic in one previous graduate survey course on IT policy). My second worry concerned the course reading load of about 250 pages per week (probably triple that of most other courses taken by our undergraduates). Could they—and would they—do all this reading? This load of reading was necessary, given the reading-and-discussion format of the course. I was also concerned about how well three-hour class discussions would work with twice as many students and teaching in a standard lecture room with fixed seats facing the front (rather than having half the number of students around a conference table).

I need not have worried about student participation. There was lively discussion that typically had to be cut off after ninety minutes so we could turn to other things. I attribute this

interest to three things. First, more than almost any issue other than peer-to-peer file-sharing, the students had an abiding personal interest in privacy and security issues. These were issues that touched their everyday lives, and they could see the relevance of the material to them. Second, just as I had done in the Internet and Society course, I had the students write a question for discussion in class each week. Whereas the graduate students in the Internet course had been mainly concerned with finding some question of significant intellectual interest, the undergraduates in the security course were writing up questions of personal interest. For example, as part of their personal development, many were struggling to figure out what behavior was ethical in the use of their hacking skills or their downloading of copyrighted music files. Third, in the Internet course we had talked some about current events and this had been well received; so I decided to make it a larger part of the security course. I used one of the first few class meetings primarily to discuss current events. After that, we spent a few minutes almost every class on things that had happened during the previous week. Students began on their own initiative to email me online news stories related to privacy and security, typically two or three each week; and I would circulate printouts of these stories in class for them to multi-process while we were having discussion or they were listening to a lecture.

Pairs of undergraduates were expected to lead class for forty-five minutes once during the semester. The idea was that I would lead the class time devoted to social, economic, political, and legal issues, and they would be responsible for covering some related technical topic that I had assigned (such as methods for establishing their own public key encryption, or how to wardrive or protect against it). (The topics are listed in Appendix C.) The student presentations did not work as well as I would have liked. On the positive side, many students had polished PowerPoint presentations, often supplemented by interesting and relevant video clips. However, the students typically were only able to sustain a twenty-five-minute period rather than forty-five, sometimes their technical coverage was not deep enough, and they had trouble moving from their presentations to engaging the class in discussion or some other exercise. I find this last point quite interesting. Although the students make abundantly clear to faculty their strong preference for active learning styles over the passive learning of lectures, when asked to cover material in any way they wish to their fellow students, they all chose the lecture format. This is presumably because an active learning style requires greater familiarity with the material, familiarity with methods for effective active learning, and confidence in one's ability to handle the course in whatever unpredictable direction it might take.

As for the use of history in this course, near the beginning of the semester we read Robert Ellis Smith's book, *Ben Franklin's Web Site* (Providence, RI, 2000). This is one of the very few publications that provides a sweeping history of privacy in America. Written by the long-term editor of *The Privacy Journal*, it is chock full of interesting historical tidbits. Unfortunately, it is highly idiosyncratic in its choice of topics and jumps around chronologically. The students were fascinated by the material, as I was, but they complained about the book being difficult to follow. It was hard to use the book as a foundation for historical lessons about privacy.

The cybersecurity field does not have a long history. The students did learn about the histories of hacking, cracking, encryption, and information warfare from Dorothy Denning's popular book, *Information Warfare and Security* (New York, 1999). However, they were turned off by Denning's textbook style, wanting instead a more monographic approach with a central thesis. The reaction was strongly favorable when the students read Steven Levy's journalistic history of cryptography and several biographical accounts of famous hackers, although these books lacked the academic rigor that many social scientists might want in their class readings. Were I to teach this course again, I might add historical literature on World War II as a way to

examine the trade-off between national security and civil liberties. The historical materials were particularly effective in getting the students to see the relationships between hacking and national infrastructures such as the telephone and electric power networks. Later in the semester, when we read the dry and abstract National Research Council's policy reports on cybersecurity and national infrastructures, the students had real-life stories to help them understand the national risk in a concrete way.

Conclusion

In the take-home final examination, in addition to an essay question on the various ways that privacy and security are related to one another, the students were asked to analyze, using the material we had discussed in class, one current event and one piece of proposed federal legislation on privacy or security of their own choosing. They were enthusiastic about the assignment because it continued to allow them to personalize their class work to their own interests, and because they believed that this assignment was helping them to hone practical skills involving what was going on in the world around them.

Informatics is a very presentist-oriented, professionally oriented discipline in which most students and faculty are more concerned with "what now" and "what next" than they are with reflecting on how we got to where we are. It is considerably easier to apply other social sciences than it is to apply history in the informatics curriculum because these other social sciences can more readily be applied to contemporary or recent phenomena that are familiar and of interest to the students. To the degree that history is of interest to these students, it is applied history that interests them—what lessons can we learn from the past that can be applied to situations we face today. Students have a much better time with historical material when it concerns a subject they can relate to—witness the much better reception of material on the history of personal computers and the Internet than to the history of the punched card machinery or mainframe computers.

Nevertheless, I feel it is part of the student's professional training to learn about the heritage of their discipline; and I also believe that historical case studies can be effective in teaching contemporary topics about the relationships between people, organizations, and information technologies. In these courses on social informatics, however, I carefully limit the amount of historical material that I teach to no more than about fifteen percent of my course material—sometimes much less. For these same reasons, I have not taken up the offer to teach a history of computing course at Indiana University, even though the challenge of developing a new course focused on applications and context—in order to fit the Informatics approach—is appealing.

William Aspray is Rudy Professor of Informatics at Indiana University in Bloomington. He also holds faculty appointments in the computer science department, the history and philosophy of science department, and the School of Library and Information Science. He has served as Associate Director of the Charles Babbage Institute for the History of Information Processing, director of the IEEE Center for the History of Electrical Engineering, and executive director of the Computing Research Association. He has also taught at Harvard, Penn, Rutgers, Virginia Tech, and Williams. He serves on the history committees of ACM, IFIP, and SIAM. Current scholarship is focused on policy and historical issues relating to information technology. Author's address: School of Informatics, 901 E. Tenth St., Indiana University, Bloomington, IN 47405 USA. Email: waspray@indiana.edu.

Appendix A

Informatics 303 Organizational Informatics (Spring 2003)

Course description

This course examines the various needs, uses, and consequences of information in organizational contexts. Topics include organizational types and characteristics, functional areas and business processes, information-based products and services, the use of and redefining role of information technology, the changing character of work life and organizational practices, sociotechnical structures, and the rise and transformation of information-based industries.

Course organization

The use of information systems to carry out work in various kinds of organizations will be the general organizing theme for the course.

The course has three components:

1. Lectures on various topics in organizational informatics, including:
 - a. Information science analysis of electronic recordkeeping systems
 - b. Business history analysis of the origins and structures of the computer and related industries
 - c. Sociological analysis of the acceptance and use of IT in organizations
 - d. Economic analysis of the management and use of IT in organizations
 - e. Policy analysis of the IT workforce
 - f. Organizational management analysis of the uses of information systems in various industries and in the non-profit sector
2. Guest lectures by practitioners presenting real-world perspectives on the use of IT in organizations
3. Student self-study of the basic principles of business information systems

Required Textbook

The assigned textbook is Steven Alter, *Information Systems: The Foundation of E-Business* 4th ed., Prentice-Hall, 2002. Although the book is expensive, you are strongly encouraged to buy your own copy. Make sure that you buy the 4th edition! We will be reading the entire book.

Quiz and Exam Policy

Material to be tested on quizzes or exams: all material in the assigned readings, plus all material covered in class by the instructor or guest lecturers

Quizzes: We will only count 10 of the 13 quiz scores. Thus if students miss three or fewer quizzes for reasons of illness or religious observance, makeup quizzes will not be given. If a student misses more than three quizzes for reasons of illness or religious observance and can provide written documentation of their reasons, makeups will be given for the number in excess of three that are excused misses. For any student who has taken more than ten quizzes, the ten highest grades will be counted.

Exams: In-class Exams 1 and 2 and the Final Exam.

Why we give so many quizzes and exams:

1. Testing makes you keep up with the material; and students who keep up with the material learn it better than those who cram.
2. Testing reinforces your understanding of the material –you have an active rather than a passive knowledge of the material.
3. The amount of material you need to learn for any one test is reduced –making your study efforts easier.
4. A few poor performances on quizzes (low scores, missed quizzes) can be discounted because we recognize that students do have occasional times when they can not pay full attention to the course (excessively busy with another course, ill, away on job recruitment or family emergencies, etc.);
5. No one quiz or exam counts that much towards the final grade, so a poor performance in one test can be overcome.

Grading and Student Misconduct

The course grade will be determined as follows:

- 25% Exam 1
- 25% Exam 2
- 25% Quizzes (adjusted for participation in discussion section)
- 25% Final Exam

There will be one extra-credit assignment. Details will be discussed in class.

Class schedule:

1. M Jan 13 Course Organization, What Organizational Informatics Is
2. W Jan 15 Computer Industry I (Predecessor Businesses)
3. F Jan 17 discussion –no quiz
4. M Jan 20 no class Martin Luther King Day
5. W Jan 22 Guest Lecturer Philip Bantin (University Archivist, IU) - Electronic Records I (Basic Concepts)
6. F Jan 24 discussion –Quiz: Alter Ch. 1 (E-Business)
7. M Jan 27 Computer Industry II (Mainframe Computers)
8. W Jan 29 Guest Lecturer Jennifer Kurtz (Director eCommerce Office, Indiana Department of Commerce)
9. F Jan 31 discussion –Quiz: Alter Ch. 2 (Business Systems)
10. M Feb 3 Computer Industry III (Mainframes continued)
11. W Feb 5 Guest Lecturer Dennis Morrison (CEO, Center for Behavioral Health)
12. F. Feb 7 discussion –Quiz: Alter Ch. 3 (Business Processes)
13. M Feb 10 Guest Lecturer Philip Bantin –Electronic Records II (Recordkeeping Systems)
14. W Feb 12 Guest Lecturer Julia Sutherland (Director, Technical Services, Monroe County)
15. F Feb 14 discussion –Quiz: Alter Ch. 4 (Databases)
16. M Feb 17 Computer Industry IV (Minicomputers and Microcomputers)
17. W Feb 19 Guest Lecturer Mark Bruhn (Chief IT Security and Policy Officer, IU)
18. F Feb 21 discussion –Quiz: Alter Ch. 5 (Information System s)
19. M Feb 24 Computer Industry V (Software and Services)
20. W Feb 26 EXAM 1 (covers material January 13 through February 17)

21. F Feb 28 discussion –Quiz: Alter Ch. 6 (Customers and Products)
22. M Mar 3 Computer Industry VI (Internet)
23. W Mar 5 Guest Lecturer Alan Minton (Cornerstone Info Systems)
24. F Mar 7 discussion –Quiz: Alter Ch. 7 (Ethics)
25. M Mar 10 IT Workforce
26. W Mar 12 Guest Lecturer Steve Erlich (Aprimo)
27. F Mar 14 discussion –Quiz: Alter Ch. 8 (Networks)
28. M Mar 17 no class –Spring Break
29. W Mar 19 no class –Spring Break
30. F Mar 21 no discussion –Spring Break
31. M Mar 24 Economics and Business I (Productivity Paradox - the Classical Problem)
32. W Mar 26 Guest Lecturer Marlin Eller (SunHawk)
33. F Mar 28 discussion –Quiz: Alter Ch. 9 (Software)
34. M Mar 31 Economics and Business II (Productivity Paradox - Recent Scholarship)
35. W Apr 2 Guest lecturer Herb Senft (1stBooks)
36. F Apr 4 discussion –Quiz: Alter Ch. 10 (Telecommunications)
37. M Apr 7 EXAM 2 (covers material from February 19 through March 31)
38. W Apr 9 Sociology I (Case Study –Alpha Corporation)
39. F Apr 11 discussion –Quiz: Alter Ch. 11 (Information Systems Planning)
40. M Apr 14 Sociology II (Zeta Corporation)
41. W Apr 16 Guest Lecture - Martin Siegel (IU and WisdomTools)
42. F Apr 18 discussion –Quiz: Alter Ch. 12 (Building Information Systems)
43. M Apr 21 Sociology III (Zeta Corp. cont.)
44. W Apr 23 Extra-credit presentations
45. F Apr 25 discussion –Quiz: Alter Ch. 13 (E-Business Security)
46. M Apr 28 TBD
47. W Apr 30 Remaining extra-credit presentations
48. F May 2 discussion –no quiz - Final Exam Review
49. W May 7 (12:30 –2:30) FINAL EXAM

Appendix B

Informatics 400/590 Internet and Society (Fall 2004)

Course Description

This course will cover the origins, governance, economics, operation, and use of the Internet.

Governance and economic issues include who makes decisions about domain name and spectrum allocation, technical standards and innovation (IPv6, innovation in an end-to-end architectural environment, open and closed standards), and how the Internet is paid for. Operational issues include spam, worms, viruses, distributed denial of service, accommodation for those individuals with various impairments, and accessibility to a wide range of individuals (Digital Divide issues). Use issues include cybersquatting and trademark infringement, peer-to-peer music- and video-sharing, copyright and copyleft, use of the Internet for vices such as gambling and pornography, e-commerce and taxation, use by the government for e-voting and improved government services, and privacy considerations. Although some technical issues will be discussed, the focus will primarily be on social, political, legal, economic, and historical examination.

Admission requirements: Junior, senior, or graduate school standing, enrollment is limited to 25.

Books to Buy (for both Informatics 400 and Informatics 590 students).

Janet Abbate, Inventing the Internet (MIT Press, 2000) paperback

Milton Mueller, Ruling the Root: Internet Governance and the Taming of Cyberspace (MIT Press, 2002)

Lawrence Lessig, Code and Other Laws of Cyberspace (Basic Books, 2000) paperback

Lawrence Lessig, The Future of Ideas: The Fate of the Commons in a Connected World (Vintage, 2002) paperback.

Jessica Litman, Digital Copyright: Protecting Intellectual Property on the Internet (Prometheus, 2001)

Philip Jenkins, Beyond Tolerance: Child Pornography Online (NYU Press, 2001)

Shanti Kalathil and Taylor Boas, Open Networks, Closed Regimes: The Impact of the Internet on Authoritarian Rule (Carnegie Endowment for International Peace, 2003)

Supplemental Reading:

For those who want to know more about the technology of the Internet, pick up copies of the various books by Simson Garfinkel or Gene Spafford. For those who want a primer in legal issues, look at www.nolo.com.

Weekly Topics and Reading Assignments:

1. September 2: Course Introduction

2. September 9: Digital Divide and Americans With Disabilities Act

400 students: read Amanda Lenhart et al., "The Ever-Shifting Internet Population", The Pew Internet & American Life Project, April 2003; National Council on Disability, The Accessible Future Washington, DC, June 21, 2001; National Telecommunications and Information Administration, "Falling Through the Net: Toward Digital Inclusion" (October 2000). This report and the three previous reports in this series can be found online at <http://www.ntia.doc.gov/ntiahome/digitaldivide/index.html>

590 students: read Lenhart et al.; National Council on Disability study; NTIA 200 study; some readings of your choice from Benjamin Compaine, ed. The Digital Divide (MIT Press, 2001) or other sources of your choice on the Digital Divide as it relates to the Internet.

3. September 16: History of the Internet

400 students: read Abbate Ch. 1-4, 6

590 students: read Abbate entire book

(If you do not wish to buy a copy, the IU libraries has a copy available online that you can download.)

4. September 23: Laws, Norms, Markets, and Architectures

400 students: read Lessig, Code ch. 1-8, 15

590 students: read Lessig, Code entire book

5. September 30: Governance, Standards, Economics, and Innovation

400 students: read Mueller ch. 1-9

590 students: read Mueller entire book

6. October 7: Spam, Worms, Viruses, and Distributed Denial of Service

400 students: Brian Krebs, "A Short History of Computer Viruses and Attacks", washingtonpost.com, 14 February 2003; Edward H. Freeman, "Prosecution of Computer Virus Authors," Legally Speaking, March/April 2003, pp. 5-9; Statement of Eric Holder Before the Subcommittee on Crime, "Internet Denial of Service Attacks and the Federal Response," February 29, 2000

590 students: Krebs; Freeman; Holder; plus any one of the following three: John Eisinger, "Script Kiddies Beware: The Long Arm of U.S. Jurisdiction to Prescribe," [Washington & Lee Law Review](#) 59 (Fall 2002); readings in Kevin Mitnick's book on hacking; or do some online web browsing and bring some recent stories on this subject to class.

7. October 14: Intellectual Property 1: Patents and Trademark Issues (Business Process Patents, Cybersquatting, etc.)

400 students: read Litman ch. 1-5

590 students: Litman ch. 1-5; Janell Kurtz and

Cynthia Mehoves, "Whose Name is it Anyway?" [Marketing Management](#) Jan/Feb 2002, pp.30-34.

8. October 21: Intellectual property 2: Copyright Issues (Peer-to-peer music and video networks, copyleft, creative commons)

400 students: read Litman ch. 6-13,

590 students: read Litman ch. 6-13; in addition, do one of the following two tasks: look online at material about both the GNU General Public License (copyleft) and Creative Commons; or read some parts in Siva Vaidhyanathan, *Copyrights and Copywrongs: The Rise of Intellectual Property and How it Threatens Creativity* (New York University Press, 2001).

9. October 28: guest lecture by Prof. Christine Ogan (Informatics and Journalism) –Internet confession sites. Readings, if any, to be determined.

10. November 4: Digital Government and E-voting

400 students: read Kevin Coleman, "Internet Voting" (CRS Report for Congress, Updated January 31, 2003, Congressional Research Service, Library of Congress); Rebecca Mercuri, "A Better Ballot Box?", *IEEE Spectrum* October 2002, pp. 46-50.

590 students: read Coleman; Mercuri; plus "Voting: Not Yet on the Net", *Minnesota Planning* November 2002, pp. 1-10; Rebecca Mercuri's website

11. November 11: Free speech, Pornography, Cyberstalking, and Gambling

400 students: read Jenkins entire book

590 students: read Jenkins entire book; U.S. General Accounting Office, "Internet Gambling: An Overview of the Issues" (GAO-03-89. December 2002); "Cyberstalking –A New Challenge for Law Enforcement", Ch. 1 in Department of Justice, *Stalking and Domestic Violence Report to Congress*, May 2001; Mark C. Alexander, "The First Amendment and the Problems of Political Viability: The Case of Internet Pornography", *Harvard Journal of Law and Public Policy* vol. 25, pp. 979-1030.

12. November 18: E-commerce and E-taxation

400 students: read Lessig, *Future* ch. 1-8

590 students: read Lessig, *Future* entire book

13. November 25: International Jurisdiction

400 students: read Kalathil and Boas, entire book

590 students: read Kalathil and Boas, entire book; Jack L. Goldsmith, "Against Cyberanarchy," *University of Chicago Law Review* vol. 65 (1998), pp. 1199-1250; David G. Post, "Against 'Against Cyberanarchy,'" *Berkeley Technology Law Journal* vol 17 (2002), pp. 1365-1387.

14. December 2: guest lecture, Prof. Susan Herring (SLIS), Internet, gender, and communication –class this week runs 5:00 –7:00 PM instead of 4:00 –7:00 PM. Readings to be determined.

15. December 9: Privacy

400 and 590 students: read some in the following sources: Robert Ellis Smith, *Ben Franklin's Web Site: Privacy and Curiosity from Plymouth Rock to the Internet* (Providence, RI: Privacy Journal, 2000); Phil Agre and Marc Rotenberg, ed., *Technology and Privacy: The New Landscape* (MIT Press, 1998); Fred H. Cate, *Privacy in the Information Age* (Washington, DC: Brookings Institution, November 1997).

Assignments

Classroom presentations

Each I400 student will be asked to make a 15-minute classroom presentation on a topic assigned by the instructor. Each I590 student will be asked to make two 15-minute presentations on a topic assigned by the instructor.

Weekly written assignments

Each week (except for weeks 1, 9, and 14) the student is expected to prepare a single question for in-class discussion that week. The top 10 of 12 scores will be counted as part of the final grade. The kinds of questions and how to prepare them will be discussed in week 1. The questions may not be more than one page in length. They must be in the instructor's hands by noon on class day, sent electronically to waspray@indiana.edu. Papers not in my electronic possession by noon will not be accepted. Grading will be as follows:

- 0 - no paper or late paper or unacceptable quality
- 1 - question is on topic but shows limited insight or contains errors
- 2 - question is entirely acceptable but not selected for class use
- 3 - question is accepted for class use but modified by instructor
- 4 - question is accepted for class use without modification

Final assignment

I400 students will have an in-class, open-book, open-note essay exam at the regularly scheduled final exam time.

I590 students will have to prepare a five-page essay due at the time of the regularly scheduled final exam. Details about content will be given later in the semester.

Class discussion

Every student is expected to come to class having read the assigned material carefully and ready to participate in the discussion. Class participation will be factored into the final grade.

Appendix C

Informatics 400/590 Privacy, Security, and Information Warfare (Spring 2004)

Course Description

This course will examine issues of computer privacy, computer security, and information warfare. The focus will mostly be on the Internet, but attention will also be given to databases and to systems such as the telephone and electrical networks that are highly reliant on information technology. While the course will not neglect technical issues, the focus will primarily be on political, social, economic, legal, managerial, and regulatory issues. Topics that we will likely cover include authentication, filtering, encryption, biometrics, cookies, hacking and cracking, employer surveillance, spamming, denial of service, malicious code, cybersquatting, social engineering, the concept of trust, government intrusion versus individual liberty, computer fraud, eavesdropping, identity theft, organizational risk management, national critical infrastructures, and economic espionage. This is a reading and discussion seminar, with some classroom presentations by students and the instructor.

Books to buy for this course –These books are not being ordered by the bookstore, so you are required to order them for yourself. Remember that it may take several weeks for an order to be delivered by mail, so order early enough that you have the books in time for class.

Fred Cate, Privacy in the Information Age (1997)

Dorothy Denning, Information Warfare and Security (1998)

Amitai Etzioni, The Limits of Privacy (2000)

John Hennessy, David Patterson, and Herbert Lin, eds., Information Technology for Counterterrorism: Immediate Actions and Future Possibilities. Computer Science and Telecommunications Board (2003) [This book can be downloaded free of charge in HTML format from http://www7.nationalacademies.org/cstb/pub_counterterrorism.html]

Stephen Kent and Lynette Miller, eds., Who Goes There? Authentication Through the Lens of Privacy, Computer Science and Telecommunications Board (2003) [This book can be downloaded free of charge in HTML format from http://www7.nationalacademies.org/cstb/pub_authentication.html]

Steven Levy, Crypto: How the Code Rebels Beat the Government Saving Privacy in the Digital Age (2002)

Jonathan Littman, The Watchman (1997)

Kevin Mitnick, The Art of Deception: Controlling the Human Element of Security (2002)

Stewart Personick and Cynthia Patterson, eds., Critical Information Infrastructure Protection and the Law, Computer Science and Telecommunications Board (2003) [This book may be viewed and printed, one page at a time, in the OpenBook format at http://www7.nationalacademies.org/cstb/pub_ciip.html. It is also possible to buy a copy or buy a PDF electronic copy.]

Fred Schneider, Trust in Cyberspace, Computer Science and Telecommunications Board (1999) [This book can be downloaded free of charge in HTML format from http://www7.nationalacademies.org/cstb/pub_trust.html]

Robert Ellis Smith, Ben Franklin's Web Site: Privacy and Curiosity from Plymouth Rock to the Internet (2000)

Weekly Topics and Reading Assignments

Each class will be divided into two parts, A and B, with a break in between. Part A will typically last just less than two hours and will be devoted to discussion of the main readings of the week. Part B will typically last just less than one hour and will be devoted to examination of some technical issue.

Required reading is to be completed prior to the class for which it is assigned. You should read it carefully enough that you can actively participate in the class discussion.

I reserve the right to change the schedule at any time during the semester.

1. January 14 - Introduction.

Part A, B: Class discussion of what is privacy, security, information warfare; structure of the course; ways of theorizing (artifacts have politics, code and other ways of regulation, who makes law); how to approach assigned readings; appropriate questions for your weekly assignments; assignment of Part B presentations to students.

Required reading: none.

Optional reading: Computer Science and Telecommunications Board, The Internet Under Crisis Conditions: Learning From September 11 (2002) [You can view this book in OpenBook format, free of charge but only one page at a time, at http://www7.nationalacademies.org/cstb/pub_internet911.html]

2. January 21 - Law and policy issues of privacy: an overview

Part A: video on privacy: Jonathan Zittrain, Molly von Houweling, Stanford Law School, 2003.

Part B: Discussion of current events: privacy and automobiles; Northwest data to NASA; *Hiibel v. Nevada*; RFIDs at Walmart, Marks and Spenser

Required reading: none.

Optional reading: David Bruggeman on security, Najma Yousefi on privacy [You will find these chapters as PDF files on Oncourse. They are chapters from the instructor's forthcoming edited book entitled Chasing Moore's Law: Information Technology Policy in the United States.]

3. January 28 - History of Privacy in America.

Part A: Breakout discussions of red, white, and blue groups, followed by general discussion of Smith. Presentation on RFIDs and loyalty cards.

Part B: Student presentation: video surveillance, keystroke monitoring, and Van Eck receptors.

Required reading: Smith (entire book). Pay particular attention to how law, norms, market, and architecture are used to regulate behavior

4. February 4 - Privacy in an Online World.

Part A: Overview presentation on privacy policy in America: brief history, government as player in policy. Breakout discussions of European and US privacy policies, then general discussion of Cate.

Part B: Student presentation: cookies and privacy.

Required reading: Cate (entire book) - the most important parts are the chapters on European and US policy and on cyberspace (through p. 132). Skip or skim the rest.

5. February 11 - Privacy –A Communitarian View.

Part A: General discussion of Etzioni. We will focus on the four case studies other than the one on strong encryption. Presentation on electronic voting and privacy.

Part B: Student presentation: identity theft.

Required reading: Etzioni (entire book). Feel free to only skim the chapter on strong encryption to get Etzioni's general position since we will be covering this topic in greater depth the following week. Only skim the final chapter quickly. Read the other four case studies and the introductory chapter more carefully.

6. February 18 - Encryption

Part A: Graduate student will lead the general discussion of Levy and make presentation on privacy issues related to digital rights management.

Part B: Student presentation: public key encryption and other kinds of encryption used today.

Required reading: Levy (entire book)

7. February 25 - Introduction to Information Warfare.

Part A: General Discussion of Denning and Littman. Taped lecture on technologies that lead to new digital rights schema that raise privacy issues.

Part B: Student presentation: digital signatures and watermarks.

Required reading: Denning, pp. 1-76; Littman (entire book)

8. March 3 - Offensive Information Warfare

Part A: Presentation on airline passenger profiling and EU-US disputes over data protection and individual privacy.

Part B: Student will lead general discussion of Denning and make a presentation on privacy issues related to Microsoft's Paladium.

Required reading: Denning, pp 77-282.

9. March 10 - Social Engineering

Part A: General discussion of Mitnick. Class exercise to do some social engineering thought experiments.

Part B: Student presentation: viruses, worms, time bombs. Trojan horses, and other forms of malicious code.

Required reading: Mitnick (entire book).

Optional reading: For a critique of Mitnick, see Jonathan Littman, The Fugitive Game.

March 17 - NO CLASS (SPRING BREAK)

10. March 24 - Defensive Information Warfare

Part A: Student presentation: buffer overflows, password crackers, and other means of hacking.

Part B: General discussion of Denning and Dourish.

Part C: Student presentation: distributed denial of service attacks.

Required reading: Denning, pp. 283-424.

11. March 31 - Authentication

Part A: Student presentation: biometrics and related means of authentication.

Part B: Guest presentation by Prof. Nicole Mineo (IUPUI): HIPAA and medical privacy

Part C (if time permits, if not, this discussion will occur the following week) General discussion of Kent and Miller.

Required reading: Kent and Miller. While the entire book is assigned, the most important chapters are chapter 1 (especially the long example of authentication), chapter 4 (especially the material about user-centered design), chapter 5 (authentication technologies), and chapter 6

(especially the recent policy examples). Chapter 3 is an excellent introduction to privacy, but you will know all that material by the time you get to this point in the semester.

12. April 7 - Trustworthy Networked Information Systems

Part A: Presentation by Mark Bruhn, VP of Information Security, IU on Defensive Information Warfare at IU. General discussion of Schneider.

Part B: Student presentation: firewalls.

Required reading: Schneider: Chapters 1 through 5. Of particular importance is Chapter 2 in which the trustworthiness of the telephone and Internet networks are compared. Chapter 3 is mostly about software engineering; you do not need to understand all the technical details in that chapter, but do read enough that you understand the approach used by the software engineering community for building software that meets given specifications. Chapter 4 covers some new material and some topics we have already covered (e.g. public key encryption); feel free to skip topics we have already read about for class. Chapter 5 is about building trustworthy systems from untrustworthy components; as with Chapter 3, you don't need to know all the details, but read enough of Chapter 5 to understand the approach.

13. April 14 - Critical Information Infrastructures –Legal and business issues

Part A: General discussion of Personick and Patterson. Presentation on USA Patriot Act.

Part B: Student presentation: spam and pornography filters.

Required reading: Schneider (chapter 6); Personick and Patterson (entire book - it's short!)

14. April 21 - IT and Counter-terrorism

FINAL ASSIGNMENT DUE TODAY!

Part A: General discussion of Hennessy, Patterson, and Lin. Discussion of answers to final assignment.

Part B: Student presentation: preventing theft of service such as Blue boxes, calling card misuse, cellular fraud, and wardriving.

Required reading: Hennessy, Patterson, and Lin (entire book -its short! Pay particular attention to their lists of research projects that need to be done, as described in chapter 3 and in box 5.1)

15. April 28 - open for catch-up, new topics

Classroom presentations

Each I400 student will be asked to join with one other I400 student to make one 45-minute classroom presentation on a technical topic assigned by the instructor. These presentations should include a lecture/tutorial portion. It is also acceptable, but not required, for the presenters to have the other students do some exercises based upon the material presented,

organize discussion sections, have mock trials, or do some other activity in which the rest of the class is actively engaged. Assignments will be made at the first meeting of the class.

Each I590 student will be asked to make two presentations on topics assigned by the instructor. Some of these may be technical (Part B) topics; others may involve presenting some material or opening discussion in Part A of the class.

Weekly written assignments

Each week (except for weeks 1, 2, and 15) every student (from both I400 and I590) is expected to prepare a single question for in-class discussion that week. The top 10 of 12 scores will be counted as part of the final grade. The kinds of questions and how to prepare them will be discussed in week 1. The questions may not be more than one page in length. They must be in the instructor's hands by 10:00 AM on class day, sent electronically to waspray@indiana.edu. Papers not in my electronic possession by this deadline will not be accepted. Grading will be as follows:

- 0 - no paper or late paper or unacceptable quality
- 1 - question is on topic but shows limited insight or contains errors
- 2 - question is entirely acceptable but not selected for class use
- 3 - question is accepted for class use but modified by instructor
- 4 - question is accepted for class use without modification

Final assignment

I400 students will have a short take-home final examination. Details will be given part way through the semester.

The I590 students will work with the instructor as a team to write a research paper on a topic chosen by the instructor on the social, political, legal, or economic dimensions of privacy, security, and information warfare.

Class discussion

Every student is expected to come to class having read the assigned material and ready to participate in the discussion. Class participation will be factored into the final grade.

Religious holidays and cheating

The university has approved policies about missing class and making up work due to absence for religious holidays, as well as unacceptability of cheating and penalties for it. These policies are posted on the IU-B web pages and will be followed in this class.

6. Introducing Humanistic Content to Information Technology Students

Atsushi Akera and Kim Fortun
Rensselaer Polytechnic Institute

Abstract

This article describes a two-course sequence, offered during the first year, which constitutes a “humanities and social sciences core” for all Information Technology (IT) students at Rensselaer. This intensive exposure to humanistic thinking and social analysis is essential to developing a student’s identity as an IT professional, as distinct from computer science. “IT Revolution—Myth or Reality” uses the history of information and communications technologies to broaden our students’ perspectives about the social potential of technology. “The Politics and Economics of IT” offers students an opportunity to experience, and practice, real-world problem-solving through a large-scale entrepreneurial simulation. Both courses employ innovative pedagogic strategies to provide our students with important and practical skills for being an IT professional.

An interesting feature of this volume is that, although the NSF workshop was organized to ask, “How do we use history to teach computer science?” the historians assembled had found themselves responding to local pressure to teach students in every computer-related field *except* computer science. Over the past two decades, the dominance of computer science has given way to a more pluralistic arrangement. New schools and programs for information science, social informatics, and information technology (IT) have come to populate the academic landscape. Early exposure to the humanities and social sciences is even more important to the professional development of students in these fledgling disciplines.

While historical material can help contextualize and drive home technical lessons in computer science, an awareness of the social and historical dimensions of computing may be integral to the disciplinary practice of those entering these new fields. This paper recounts our experiences in developing what amounts to a two-semester curriculum that constitutes a “humanities and social science core” required of all IT majors at Rensselaer. We have been able to provide our students with an array of practical skills drawn from the humanistic disciplines, even in providing a curriculum that upholds a strong vision of liberal education.

Origins of Rensselaer’s IT Program

Rensselaer was among the first academic institutions in the United States to create an undergraduate and graduate degree program in information technology. Locally, the program’s administrators are very proud of this fact—as they should be—although it is also worth recounting the local circumstances that made this possible. As an engineering institution, Rensselaer has always been responsive to the employment demands of industry. Undergraduate education became all the more important amid the post cold-war contraction in research spending. At the same time, new developments in IT were generating considerable excitement. The tremendous demand for an IT workforce during the mid 1990s made it especially attractive to create a degree program in the field. This idea dovetailed with a number of other institutional priorities, including those of increasing undergraduate enrollment and retention, and dealing with the broader fiscal challenges that Rensselaer was facing during the early 1990s.

Rensselaer was also able to create the program quite quickly because of the “low walls” between its organizational units. As a mid-sized engineering school, Rensselaer lacked the

disciplinary barriers and entrenched differences that have prevented IT programs from being established at other institutions. The governance structure of the institution is such that the members of the different schools often interact with one another, and there was considerable local excitement about “interdisciplinarity.” Moreover, given that Rensselaer’s emphasis was on undergraduate education, this prevented the new IT initiative from being captured by the dictates of computer science-based research. It also helped that the computer science department was overwhelmed with enrollments at the time, and therefore expressed no serious opposition to a new degree program that would draw away some of their students.

As originally conceived, the new initiative in IT drew considerable interest from the faculty—precisely those faculty interested in new interdisciplinary research in computing-related fields. Nevertheless, it is notable that the move to create an IT program was an initiative that originated with the administration. Significantly, Rensselaer’s IT program was established as an autonomous degree-granting unit placed directly within the Provost’s office, as opposed to one of the five established schools within the institution. The new IT faculty, moreover, was drawn from existing academic departments, where these individuals continued to hold their primary academic appointments. This at least represented a structural shift in the authority of the central administration because it allowed various issues, including curriculum, to be determined at this level, rather than through a process where faculty interests and autonomy were protected through the conventional hierarchy of schools and departments. On the other hand, this ability to cut across departmental boundaries was considered essential to the interdisciplinary foundations of IT. An undergraduate curriculum committee, comprised of the faculty of the five schools, was also established to ensure that the curriculum itself was designed by the faculty and not the administration.

Indeed, given the unusual nature of the proposed arrangement, plans for the new IT program were advanced through open meetings with representatives from each of the five schools—architecture, science, engineering, management, and humanities and social sciences. These early conversations led to a consensus that one of the key things that would distinguish IT students from those trained in computer science would be their knowledge of an application domain. All students would be required to declare a “second discipline” (now called “concentration”) in which they would complete their specialization in IT. This presented an amicable solution in terms of distributing undergraduate enrollments across the five schools. Moreover, as introduced during a period of incentive-based budgeting where undergraduate enrollments were tied to departmental budgets, this created a specific incentive for academic departments to advance “second discipline” curricular templates to attract their share of IT students. The IT program was launched in 1998, providing students with a choice of twenty-one second disciplines.

What should also be noted about the undergraduate curriculum committee was the equal seat given to a representative from the School of Humanities and Social Sciences, along with those from architecture and management. Still, given the technical orientation of those who identified themselves as computing and IT professionals, it was not a foregone conclusion that the humanities and social sciences would be an integral part of the new IT curriculum. However, there were related developments that shaped the course of discussions. During the late 1990s, there was broad discussion about engineering education reform—the discussion that eventually led to the ABET 2000 requirements. While the specific influences would be difficult to track down, as an engineering school many members of the Rensselaer faculty were aware of the call to broaden engineering education by including aspects of professional development, ethics, and social understanding. The School of Engineering had in fact recently completed a major

curricular reform of its own, during which it made various requests about how its students should meet their requirements in the humanities and social sciences.

In the end, the decision was made to define the IT program with a core sequence of forty-eight course credits. Of these, twenty credits were devoted to a common technical core consisting of courses such as Computer Components and Operations; Computer Architecture, Network and Operating Systems; Database Systems; and a statistical requirement.⁵⁸ During the discussion, there was some concern that a reduced technical core would handicap our students because they would hold a “watered-down” computer science degree. However, the computing faculty who had aligned themselves with the IT program continued to proceed with the assumption that this would enable a better match between the skills of the program’s graduates and employer interests. There were others who felt that this was also an opportunity to shape the IT profession in a direction oriented toward social awareness and responsibility.

In any event, the strength of Rensselaer’s IT program depended on how well the remaining parts of the curriculum were specified. Substantial effort went into developing strong curricular templates in the “second disciplines” comprised of thirty-two credits. Meanwhile, twenty-four credits were set aside for a sequence of courses in the humanities and social sciences, and in management. Describing the sequence as it now stands, this includes the two courses required in the first year (eight credits); “Creative Design in IT” and “Economics of IT” in the second year (eight credits); and a pair of management courses on “Exploiting the Information World” and “Managing IT Resources,” generally during the second and third years (eight credits). There is also an integrated IT capstone experience required of all students (four credits in core; four credits in the second discipline). Having sixteen credits of specified humanities and social sciences courses during the first two years might seem like quite a lot. However, it was argued that it was best to provide these courses early in the curriculum, while the student’s view towards the IT profession remained most malleable. Socializing students into a specific image of the profession required intensive exposure and experience.

The opportunity to offer such an experience also emerged out of the strengths of Rensselaer’s School of Humanities and Social Sciences. While nominally the school provides an important service function in offering general courses in the humanities and social sciences for Rensselaer’s technically oriented students, most of its departments and faculty specialized in the social and artistic studies of science and technology. Rensselaer’s Arts Department, for instance, has specialized in electronic arts; the Department of Language, Literature and Communications has specialized in technical communication, and representations of science and technology in literature and new media. Meanwhile, Rensselaer’s Department of Science and Technology Studies was specifically recognized for its work on the social implications of science and technology. In the end, the STS department assumed the responsibility of developing the initial two-course sequence required of all first-year IT students.

Defining a Curriculum

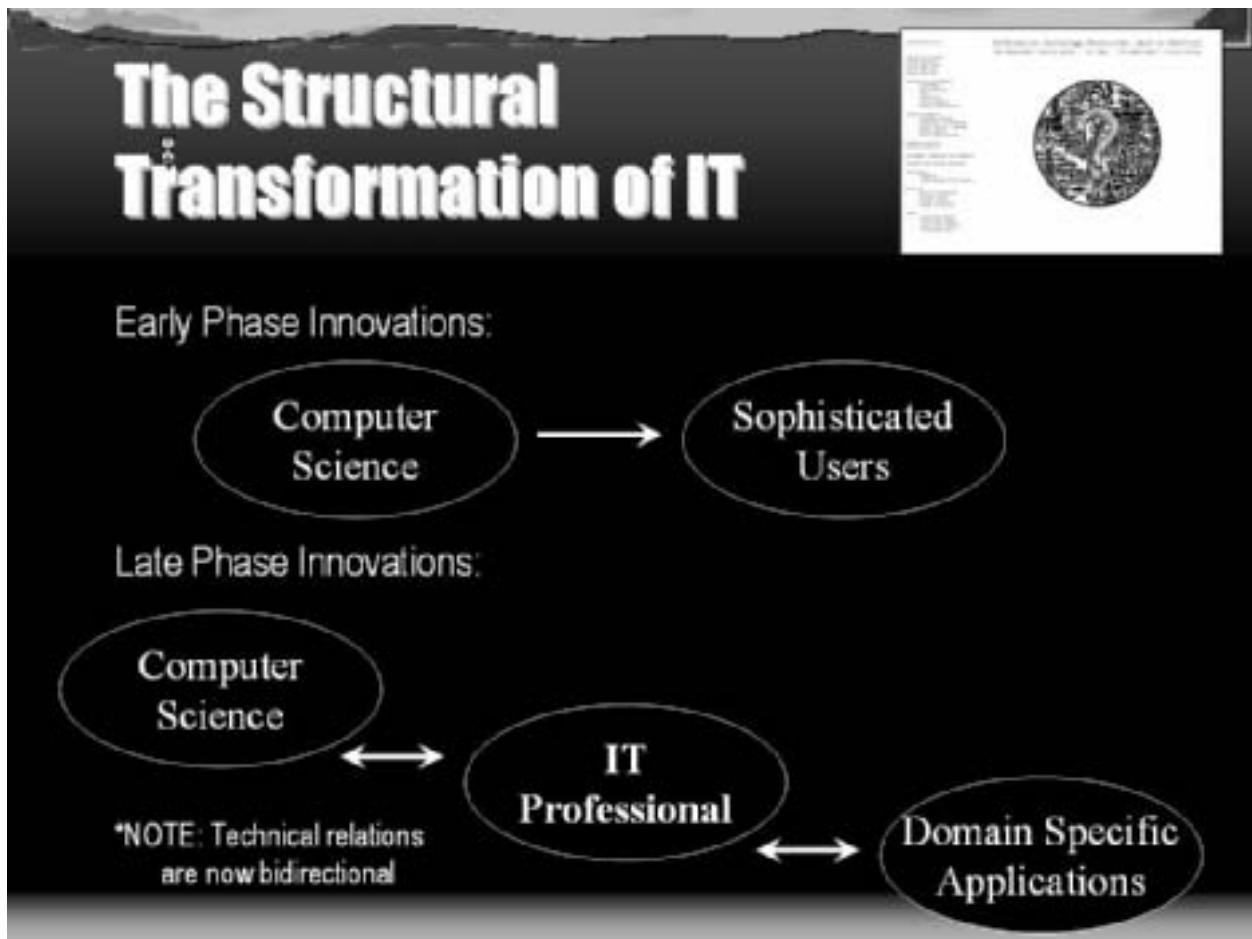
Our greatest challenge has been that of understanding our students, developing a responsible vision for the IT profession, and creating a curriculum that would cultivate the right orientation and skill set among our students. It was clear from the outset that we were being called upon to socialize our students. Most prospective IT students—seventeen- and eighteen-

⁵⁸ Students are given some options in their choice of technical core. For a precise list of current degree requirements, see http://www.rpi.edu/it/undergrad/degree_requirements.html. “Economics of IT” is a new requirement added in FY04.

year-old high school seniors approach summer orientation with little knowledge of academic disciplines, let alone of a new field like information technology. They typically choose IT as their declared major because of their interest in computers, and the fact that the phrase “IT” now appears more frequently than computer science in popular sources like *Wired* magazine.⁵⁹ Students who have some knowledge of the IT profession may choose IT over computer science because they may be more interested in developing applications, or even “working with people,” than in being a programming drone. Some also arrive with an interest in management and finance, but consider a background in IT as a unique selling point of a Rensselaer degree. Regardless, most students arrive with a predominantly technical interest with very little conceptual framework for differentiating information technology from computer science, and for understanding the different professional demands of being an IT professional.

Figure 1 is a diagram we use to illustrate the difference between computer science and the IT profession. The crude schematic obliterates important distinctions that we would want to

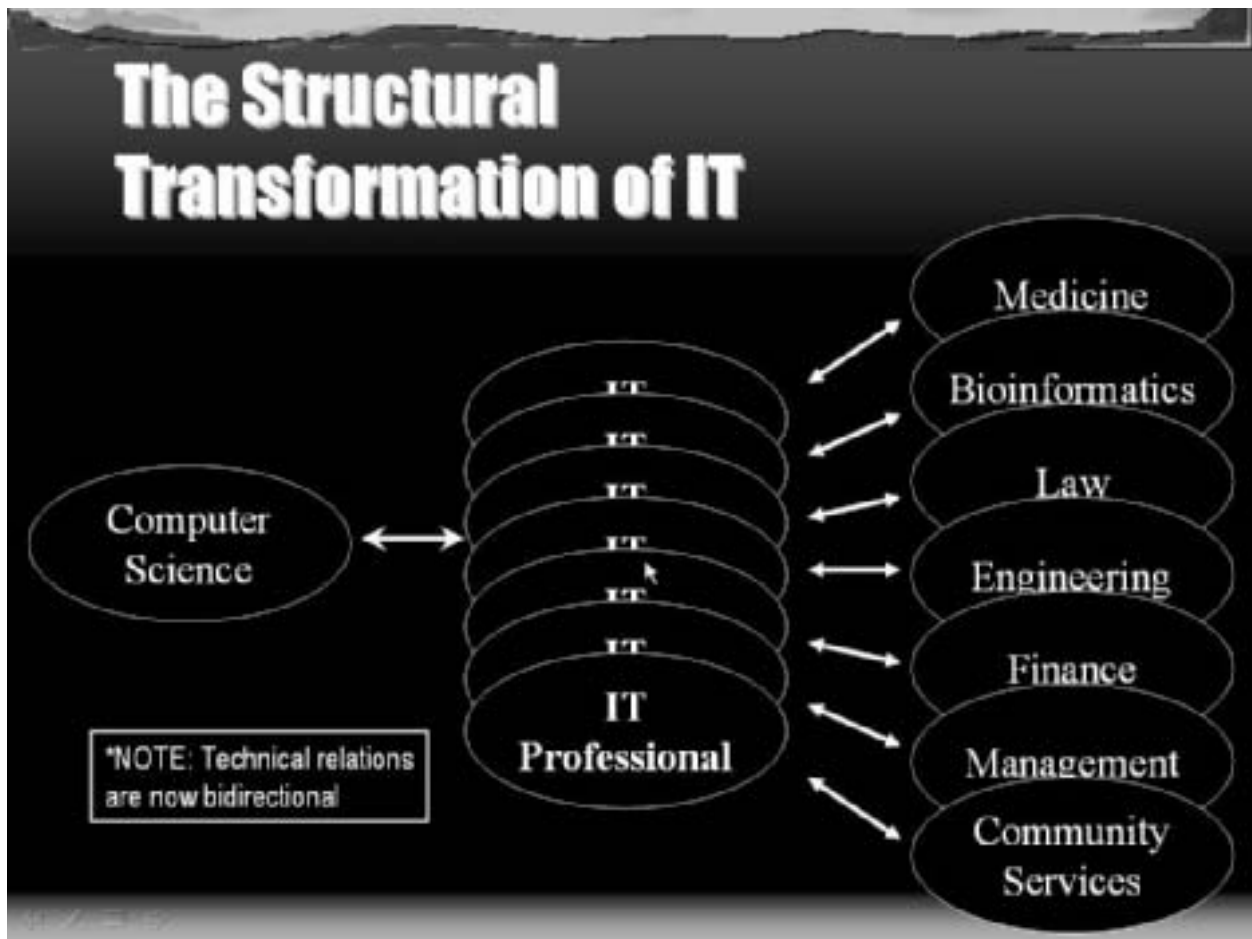
Figure 1. Slide Used to Illustrate the Structural Transformation of the Computing Field



⁵⁹ Rensselaer encourages its students to declare their major during the summer orientation period. IT students have a special incentive to declare their major from the outset because a grade point average of 3.0 is required for transferring into the program at a later date.

employ in any serious academic inquiry, but the diagram is useful for conveying several technology a trillion-dollar industry, it has become necessary to have IT professionals who are able to serve as intermediaries between computer scientists and the end-users of IT. We point out that this is why all IT students have to declare a second discipline so they are as conversant in the application domains in which they will do their work as they are with their foundations in computer science. We use Figure 2 to drive this point home.

Figure 2. A Slide Emphasizing the Role of IT Professionals as Intermediaries



We then go on to point to other subtle changes in the relationship. Whereas in the early days, computer programmers could impose a solution based on the imagined needs of the end-user, the competitiveness of the modern IT market requires software designers to be much more sensitive to the actual needs of end-users. Communications must be bidirectional, not one-way. But this suggests, then, that the skills that IT professionals have must not be technical skills alone, but ones rooted in the humanistic disciplines. We explain how the humanities and social sciences are integral to their IT curriculum, and how we were asked to develop this two-course sequence to cultivate the appropriate skills and outlook of an IT professional.

From a pedagogic standpoint the real challenge for us has been to develop a curriculum that served these objectives, while at the same time delivering the breadth of knowledge required to serve the student's need for general education. Thus, even during the early

discussions about the overall IT curriculum, it was clear that both courses should draw broadly from the humanities and social sciences in expanding the social awareness of our students. It was decided that the first course in the sequence would essentially be a “History of IT” course that would give students knowledge about the origins of their discipline, and the skills with which to relate this knowledge to contemporary issues on the social dimensions of information technology. For the second course, we decided to offer a more focused look at the contemporary political and economic dimensions of information technology, with an eye to both its productive and disruptive elements. These became “ITEC1210: IT Revolution—Myth or Reality?” and “ITEC1220: The Politics and Economics of IT.”

The specific structure and pedagogic design for each of these courses is provided separately below. (The course syllabi for both courses have been compiled separately in Part III of this volume.) However, it is worth pointing to some of the common elements.

Both courses have a heavy workload, including a final project requiring a paper of twenty-five to thirty pages. In the second course, students must also engage in various team projects that produce written work every other week; in the alternate weeks, selected teams must prepare for and participate in debates. There is a separate interview assignment, where students must set up and conduct an interview each semester. Meanwhile, students are required to submit weekly essays or writing in response to the required readings. And while the numbers may not be so impressive to those in elite, liberal colleges, we assign first-year IT students an average of sixty to a hundred pages of reading per week.

In fact, defining an appropriate reading load was perhaps the most interesting and difficult challenge. Most Rensselaer students will arrive having excelled in science and mathematics, but having never developed a similar passion for literature. Indeed, as humanities faculty, we typically forget that our students, especially our technically minded students, enter college with no orientation towards a culture of reading. While there are notable exceptions, many of our students do not pick up a newsweekly, do not read newspapers, and sailed through high school English classes by skimming through selective portions of the text. One of our explicit goals was to increase our students’ stamina for reading (and writing). Another was to foster the basic skills for reading texts critically.

The advantage of offering a required course is that we are able to insist that strong reading skills will be necessary for their professional identity as an IT professional. Nevertheless, as instructors, we have been acutely aware of our responsibility for facilitating a shift in our students’ work habits and abilities. We have thought carefully about how to structure the reading assignments, and the associated grading structure, to encourage our students to adopt a different attitude towards reading. We have drawn on a variety of popular sources such as *Scientific American*, *Technology Review*, and *Wired* magazine in acknowledging the student’s attention span and current reading abilities. These shorter readings help us introduce a diversity of perspectives, and can often be a better vehicle than academic essays for introducing critical reading skills. At the same time, popular readings often provided an essential foundation for students to engage with more academically oriented texts.

The “interview assignment” is the second common component in the curriculum. Insofar as we chose to define an IT professional as an intermediary between the computer scientist and end-user, we found it important to develop an assignment that would provide students with a basic technique for speaking with representatives of both communities. The assignment is designed more to foster an appropriate outlook, rather than fully developing interviewing skills. However, we have at least developed an interview worksheet used by the students in both

courses. In “IT Rev—Myth or Reality,” students interview an IT professional. In “Politics and Economics,” students interview IT users.

Final projects also anchor both courses so students have an opportunity to synthesize their knowledge rather than focusing on rote memorization. Indeed, both courses draw more generally on embodied learning strategies where students have an opportunity to learn by doing, rather than learning from texts. The final project in “IT Rev” and the various team projects in “Politics & Economics” require students to draw together their ideas in original form, rather than repeating what they found in their sources. The interview assignment likewise requires a student to experience the challenges of speaking with people in different and esoteric disciplines, and to see the subtleties of another individual’s work and expertise. Student presentations are also integrated into both courses, in conjunction with the final project for “IT Rev” and for all team exercises in “Politics and Economics.” All this is meant to serve the pedagogic goal of retention.

Both courses are also designed to be scalable—within limits. Given the historically fluctuating enrollment in computing-related fields, it was essential that we develop a course that could be run with significantly different enrollments. In practice, this has meant relying on a lecture-recitation format, with graduate teaching assistants leading the section meetings so as to limit short-term changes in faculty loading requirements. Both courses meet twice a week, the first hour in a large plenary (lecture) format and the second in sections with no more than twenty-five students.

We consider small section size to be an integral requirement for this kind of course. The kind of re-socialization that we hope to bring about does not come easily. Close interaction between students and their instructor is necessary to understand where the students are at any given moment, and to adjust the curriculum to be in pace with the students rather than overwhelming them with unfamiliar language and perspectives. The instructors also take advantage of a format where section meetings are held immediately after the lecture. Basic issues and rich visual materials (for instance, video segments of protests against economic globalization or current concerns about the outsourcing phenomenon) are introduced during the lecture so they can be discussed and developed in section.

We have found that seminar-style discussion was not a skill that we could assume the students had when entering the program. To the extent to which team-oriented projects are integral to the IT field, we felt that cultivating a student’s ability to express and articulate complex positions was itself an important skill to confer. Small section sizes have also proved useful for identifying students who are having difficulty making the transitions we expect, and for us to work more intensively in creating a viable option for the successful progression of these students through the program. Given our reliance on graduate teaching assistants (TA), we have held weekly planning meetings as well as an initial orientation session. We have also attempted to maintain a healthy mix of TAs who did and did not teach the course previously, so as to ensure substantial continuity at this level.

There are other ways in which the two courses either share a common pedagogy or complement each other. Both courses place significant emphasis on written and oral communications. The first course emphasizes the past, and the second, the present. The first emphasizes individual work and responsibility; the second group, work and teamwork. As we have come to develop the courses, the first course has focused more on knowledge, the second on the practice of doing IT. The assignments issued in the second course also tend to build on the knowledge, skills, and perspectives acquired in the first, and are generally more integrative

in nature. In all of the assignments, from the interviews to the critical reading exercises, we have tried to develop exercises that would allow our students to develop tangible skills that they could apply outside of the courses and in their work as an IT professional.

We should also mention that the two courses described below are neither the courses as they were initially offered nor as they currently stand. One of the co-authors of this paper, Kim Fortun, was responsible for the initial development of both courses. Since then, the teaching team has been expanded to include Kate Boyer, an historical geographer, and Atsushi Akera, an historian of technology. All of us have been very interested in pedagogic experimentation and innovation, and have benefited from each other's ideas and energies. Partly as a matter of historical record, and the continued changes to the courses, we decided that it would be best to begin by presenting a kind of "idealized" curriculum from the point of view of the authors of this paper.

ITEC1210: IT Revolution—Myth or Reality?

As the first course in the sequence, "IT Rev" bears the brunt of the burden of convincing IT students that the humanities and social sciences are integral to their future identity as an IT professional. This can be a challenge.

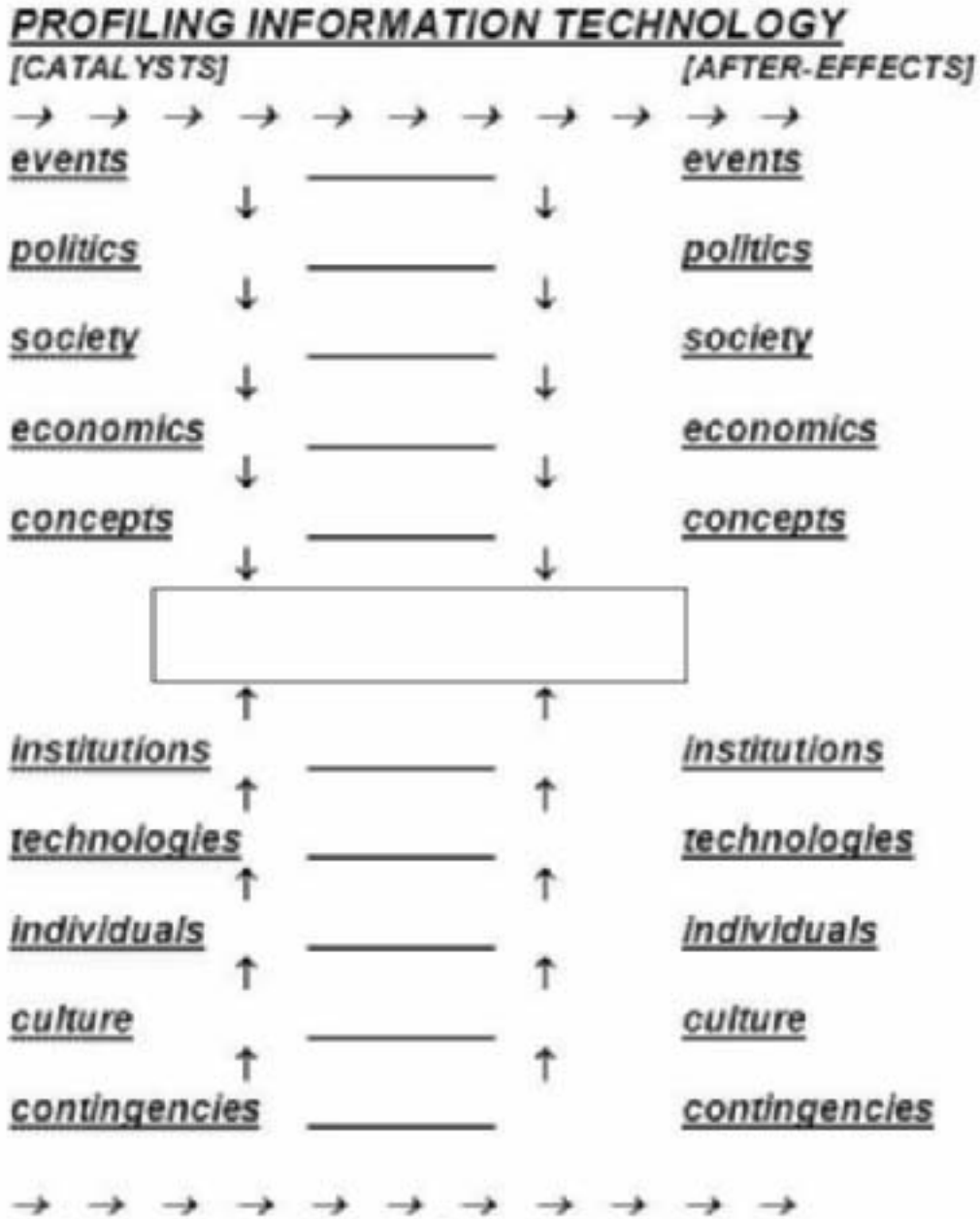
The course itself is basically a history of communications course, making it the more immediately relevant of the two courses to the specific theme of this volume. As indicated by its title, the course specifically sets out to challenge the students' assumption that the current "IT Revolution" is the most radical and innovative breakthrough in the history of humanity. The course therefore introduces students to a long succession of information technologies that contributed to radical changes in society. It covers the great voyages of the Portuguese Caravelle and the rise of capitalism and empires; the printing press and the Protestant reformation; telegraphy and its consequences for military strategy; U.S. railroads and the rise of national markets; and the rise of both broadcast radio and mass entertainment.

In doing so, we transform the question of whether IT is revolutionary to the more interesting question of what makes an IT revolution revolutionary. In fact, midway into the semester, we ask our students, "Just what kind of revolution is the IT revolution? Is it more like the French Revolution or the Industrial Revolution? What are the distinguishing features of these two archetypal revolutions, and which aspects are reflected in different developments even within the field of information technology?" In taking this approach, the course recognizes that most students will emerge from the course wedded to the idea that the current developments in IT that they are witnessing are revolutionary. Nevertheless, by transforming the question, students are asked to engage precisely in the kind of thought process conducive to critical thinking and social analysis.

In order to aid this process of critical inquiry, we have developed a "profile template" (see Figure 3) that requires students to tease apart the various social, cultural, and technical developments that contributed to or followed from the development of a technology. This diagram is used at the start of every lecture where a particular technology described during that class section is written into the middle of the diagram. Then, throughout the lecture, the specific (aspects of) events, politics, society, economics, concepts, institutions, technologies, individuals, culture, and contingencies that served as "catalysts" or "aftereffects" of the technology are discussed and added to the chart. From a purely academic standpoint, the categories employed here are not necessarily defensible; there is no simple way to distinguish "culture" from "society," and such a process of classification will tend to reduce history to a set of

operative “factors” rather than a study of their integration. However, from the standpoint of pedagogy, this exercise introduces students to the notion of multiple causation. It also is a valuable means of providing students with a way to visualize what they *do not know*. To be able to get a student to recognize that they do not yet know the cultural, economic, or political dimensions of a technology is to bring them a step closer to a liberal vision of education.

Figure 3. The Profile Template Used in IT Revolution—Myth or Reality?



Meanwhile, we help the students to draw the connection between past and present, and to exercise their skills for critical analysis, through the structure of the reading assignments. Thus, instead of assigning strictly historical texts, we match each unit with several texts that deal exclusively with contemporary issues that parallel those raised in the historical case study. Therefore, the unit on the Caravelle and early globalism is paired with articles describing economic globalization as it is unfolding today. It helps that there are many articles that place information technologies at the heart of recent changes in the global economy. Likewise, the railroad is likened to the Internet and eCommerce; telegraphy to GPS systems and other IT-based changes to military tactics and strategy; ethnic radio stations to the various subcultures currently found on the Internet; and the Internet to various visions for establishing a more popular and participatory form of governance.

We have been fortunate in that the editors of the popular science periodicals that we employ have been fascinated with the idea of comparing the past to the present. As a consequence, we have found many useful articles such as the *Wired* (April 1999) magazine piece, "I, Monopolist,"⁶⁰ which provides a simulacra of an interview with John D. Rockefeller in order to draw the obvious parallels between an early monopoly and the current business strategies of Bill Gates. There are also a number of more serious monographs where the author has used historical material to make a specific point about contemporary IT-related phenomena. Robert Verzola's "Globalization: The Third Wave,"⁶¹ posted on the *Corpwatch* website, and Anthony Walton's "Technology versus African Americans"⁶² are valuable essays that fall in this category. Meanwhile, there are a number of academic and quasi-academic texts that draw explicit parallels between the past and the present. These include Tom Standage's *The Victorian Internet* (New York, 1998) about the culture of early telegraphy, and Susan Douglas's *Inventing American Broadcasting 1899-1922* (Baltimore, 1987). Some of these texts, though not Douglas's, do present the historiographic problems of reading the past in terms of the present. However, as a pedagogic tool, the explicit comparison between the past and present appears to guarantee student engagement. It also fosters certain elements of critical thinking, especially in encouraging the kind of lateral thinking necessary to generalize beyond a single situation.

To encourage another level of critical thinking, and to hold students accountable for doing the readings, we tried using the WebCT platform and its online bulletin board capabilities. Although we do not currently use this tool because of the failings of this approach, it is worth recounting the experience. For each week's assigned readings, the instructor posted several open-ended questions on the assigned readings, each of which were designed to foster a thread of conversation. Students were required to log on to WebCT and make at least one post in response to a question or an ongoing dialogue. The exercise was attractive from the standpoint of encouraging critical engagement with the texts, and to learn a different style of conversation and argumentation than that of classroom discussion—yet another interesting skill. It also provided what seemed like a viable alternative to assigning weekly essays, which we felt would increase the workload to the point of impinging on other work for the class. In practice, however, students read selectively simply to satisfy the "WebCT" requirement. Contrary to our expectations, many of our students chose also to simply mirror the other students' ideas. As many as two-thirds of the students found practical "work-arounds" for the WebCT assignment.

⁶⁰ <http://www.wired.com/wired/archive/7.04/rockefeller.html>.

⁶¹ <http://corpwatch.radicaldesigns.org/article.php?id=1569>.

⁶² *Atlantic Monthly* (January 1999). <http://www.theatlantic.com/issues/99jan/aftech.htm>.

While we have not ruled out WebCT as a pedagogic resource, it will be necessary to develop a grading structure, and possibly a more direct process of facilitation, to make it work as we intended.

In IT Rev, we have also used class participation grades, a mid-term, and a final exam to encourage critical engagement and to hold students accountable for the readings. Class participation is especially useful to foster critical conversation skills; students are told at the beginning of the semester that attendance alone does not constitute class participation. While the use of exams is very “conventional,” it can be a useful means of conveying to technically oriented students that concrete facts are as important as critical inquiry. On the other hand, exams tend to encourage students to cram, with the attendant loss in the extent to which they bring their knowledge of the readings to the conversations in class. Rensselaer students tend, on average, to be very responsive to grade-based incentives. As a consequence, we have tended to think very carefully about how to develop a grading structure suited to the pedagogic objectives of the course.

Aside from class participation, WebCT, and the two exams, the major requirements of the course consisted of the interview assignment and a final “profile” project. For the interview assignment, students are required to interview an IT professional—originally specified as a faculty member affiliated with the IT Program. In addition to serving the stated goal of teaching students how to solicit information about the more technical dimensions of IT, the exercise was designed to introduce students to the working world of an IT professional. Sample questions were carefully designed for the students so that they would ask about the daily work routine of IT professionals, their perceptions of IT as a discipline, and the social and ethical vision that they held about their occupation. The interviews, as originally conceived, also provided an opportunity for students to introduce themselves to IT faculty members. Given that all IT faculty retain their primary commitments to other academic departments, this provided an important opportunity for IT students to interact with Rensselaer faculty.

For the final project, students were required to develop their own “profile” of an information technology of their choice—anything from a turntable, to a PDA, to an historical artifact like the astrolabe. This assignment, in particular, drew on Fortun’s background in cultural anthropology, and the concept of synchronic and diachronic analysis—thick descriptions either of a moment in time or of changes that occur through time. The assignment is based on the same profile template used every week in lecture (see Figure 3). For the synchronic analysis, the students were asked to think laterally by identifying the various social, cultural, and technical developments that were coterminous with the invention of a particular technology. For the moment, they were asked to do so without regard for whether these other events and circumstances contributed to the development or aftermath of their technology. The exercise is simply designed to stretch a student’s conceptual horizons. For instance, reading about changes in the use of Papal decrees, or the expanding pattern of trade during the invention of the printing press, could prompt students to think more broadly about social contexts and social processes as they ultimately impinge on the process of innovation.

Meanwhile, the diachronic analysis constitutes the major part of the assignment. In it, students were asked to closely follow the format of the template. But rather than thinking through the template, either in lecture or section, they were asked to produce a 300-word essay for each of the twenty categories. In considering a first-year student’s limited time-management capabilities, students were required to submit five of the twenty descriptions during each of four deadlines. Because of the multiple intervals of submission, most students were also able to learn “what was expected of the assignment,” and proceed to a deeper level of engagement

with historical material. Also, by using a grading structure that encouraged substantial revisions, we were able to meet the pedagogic goal of strengthening our students' writing skills and their ability to develop and refine their arguments.

In any event, "IT Revolution—Myth or Reality?" is a course that makes use of history in a way that meets the goals envisioned by the organizers of the NSF-sponsored workshop. At the same time, it pushes beyond the boundaries of a conventional history course in making historical material speak to current interests. It is also used to broaden student perspectives in a way that we feel is essential for the IT students in our program. A more detailed account of our assessments and experiences is provided in the conclusion to this article.

ITEC1220: The Politics and Economics of IT

A complementary set of content has always been at the foundation of the second course, "The Politics and Economics of IT." However, as the course has developed, it has moved progressively towards greater emphasis on the practice of "doing IT." Again, this is not to say that content is unimportant. The course continues to offer the students a broad set of material on subjects ranging from the digital divide, to modern medical technologies, to IT and the environment, to current concerns about corporate control of the Internet. It also introduces students to current issues concerning IT, privacy, and civil society, as well as Lawrence Lessig's latest thoughts about cyberlaw and intellectual property. Still, the course as a whole has moved from the general mode of lectures to a large-scale exercise in entrepreneurial simulation—what is referred to as educational simulation in the undergraduate pedagogy literature. We really do this on a rather large scale, with as many as sixteen teams of up to thirteen students each. Nearly all of the course exercises, apart from the individual readings and weekly essays, are tied to a semester-long exercise.

Basically, the Politics and Economics of IT begins by offering the students a Faustian bargain. Drawing on the students' interest in IT startups and the dot-com boom, we begin by asking a dozen students to step forward to be a voluntary "CEO." In exchange for this opportunity to create a fictitious IT startup, these students have to agree that everything they do during the semester has to be tied to the notion of social advocacy. Students are therefore asked to think about how they could build a better world through information technology, and specifically to act on the behalf of others who might not have the knowledge or resources to do so. In Politics and Economics of IT, it is this requirement that opens up the interesting space for social analysis. By explicitly focusing on problems that have to do with social needs and interests, and the tough issues associated with the current climate of economic liberalization, students are introduced to the complex social, political, and economic dimensions of real-world problems. By requiring students to develop practical solutions to these problems, we also end up cultivating real-world skills for working within such a heterogeneous environment. In fact, we tell the students that the skills they acquire during the course are entirely transferable outside of a non-profit environment—what, in effect, is the tougher nut to crack.

We let the students know from the outset that this is not an indoctrination exercise in progressive politics. Social advocacy simply provides a framework for carrying out social analysis, where the problems, by their very nature, preclude exclusively technological solutions. Students are also encouraged to approach their projects from their own political viewpoints. In fact, the entrepreneurial organizations set up by the "CEOs" need not be non-profits. They may be for-profit enterprises committed to social advocacy. Proposed solutions may express a range of political viewpoints, from the "wise use" policy of conservative environmentalists, to the

radicalism of Earth First. All section instructors are also reminded of Institute policy that no evaluation may be made based on a student's political views.

A list of some of the final projects undertaken by the students provides a direct indication of the diversity of student orientations and interests:

- **Volunteer Detroit** (Oasis, Inc.): A web-based solution for matching interested volunteers with community and social services organizations in the Detroit metropolitan area.
- **Dream Computers Inc.:** A for-profit enterprise that hoped to take advantage of capital depreciation schedules by accepting corporate donations of used computer equipment that could then be refurbished and sold at discount prices to schools, libraries, and other socially responsible organizations.
- **Solerin, Inc.:** A not-for-profit organization that would develop an inexpensive operating system and client-server management software, based on open systems solutions, suitable for use by community technology centers. The company would have a for-profit consulting arm for the use of the software by other organizations.

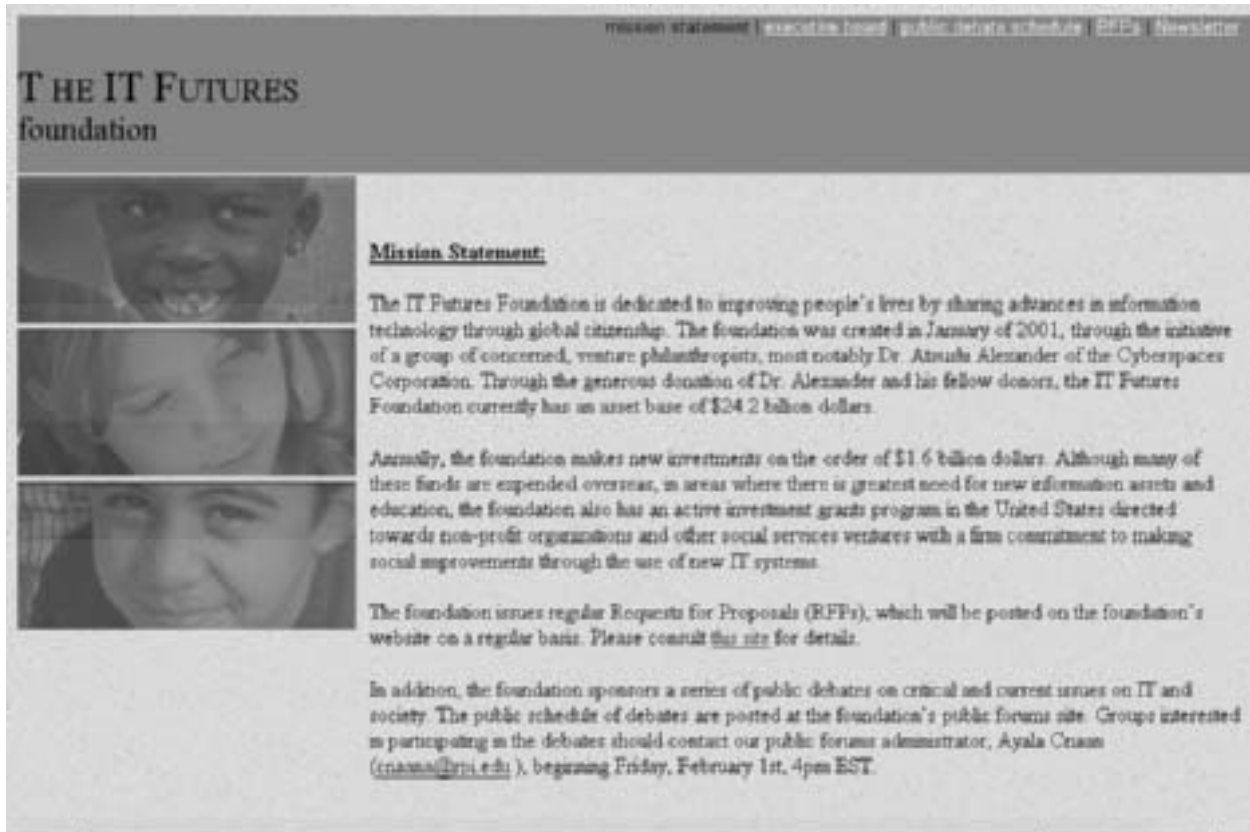
Having said this, we do find the particular inversion offered by the course to be both valuable and intriguing. At an engineering school like Rensselaer, many of the technical and managerial courses in the curriculum take economic efficiency and profitability to be a base assumption. A typical problem set in a systems engineering course will assume, for instance, that an optimized design is one conducive to the profitability of a manufacturer and the end-users. In a similar fashion, Politics and Economics of IT takes social advocacy to be a base assumption for the course. Students are therefore not asked to inquire whether social advocacy is justified in a given situation, but are given assignments that assume that social justice of one sort or another is a desired goal. Especially as contrasted against the entrepreneurial energies that drive the simulation, the apparent paradox brings many students to be quite reflexive about the proper goals of being in an IT profession. This has also provided a way of taking the present interest in "teaching entrepreneurship" and adopting it in a way consistent with the goals of general education.

What helped us set up this simulation was the popularity and visibility of IT-based philanthropy that emerged during the dot-com boom. The William and Melinda Gates Foundation, created with an initial contribution of \$17 billion, was simply the largest of a wider movement by dot-com millionaires to donate some part of their newfound wealth to a social cause. We were also fortunate to have a highly accessible *Wired* magazine article, Scott Kirsner's "The Non-Profit Motive," to get our students hooked on the idea that they could "do IT" and still make a social difference. Furthermore, the notion of venture philanthropy, as introduced in this article, turned out to be quite useful as a pedagogic tool. Insofar as venture philanthropy was about bringing the strategies of the business world into philanthropic organizations, this allowed us, as instructors, to intervene on the work produced by our students.

From a practical standpoint, what drives the overall simulation is a fictitious philanthropic organization called the "IT Futures Foundation," an organization committed to "promoting social advocacy on the Internet" (see Figure 4). The lead instructor and section instructors for the course serve as the members of the foundation's executive board. Meanwhile, students are encouraged to create their own company or non-governmental organization (NGO) by signing on other students. These other students are also encouraged to "shop" not only for a team, but the section instructor they wish to work with. The section instructors then serve, in effect, as the venture philanthropists who are willing to lend their social and entrepreneurial expertise. We

actually hold a mock IT jobs fair where the CEOs set up recruitment booths. We have had CEOs develop corporate logos, banners, and even mock-ups of their final projects—this in the third week of the class. Recruiting is allowed to begin before the jobs fair. Students are required to develop a resume outlining their capabilities and interests, and they can circulate these to the CEOs in order to schedule an interview. While team sizes are managed to reflect current enrollment levels, the course has operated with as many as sixteen teams of between eight and thirteen students.

Figure 4. IT Futures Foundation Website



Parallel to the entrepreneurial simulation is the course content and the weekly reading assignments. The first three semesters of the course are used to introduce the notions of social advocacy, venture philanthropy, social stratification, and Lessig's ideas about code and the structure of cyberspace. Weeks 4 through 14 are then organized topically around a specific concern having to do with the social dimensions of IT. The units are summarized in Table 1 below, and may be examined in more detail at the course website listed in the syllabus (see Part III of this volume). Nominally, the requirements of the course are split 50-50, so that 50 percent of the students' grades come from their individual work and 50 percent from their contributions to teamwork. However, aside from the weekly short essays, the other aspects of individual work—namely the interview assignment and "position papers"—are fully integrated with the teamwork projects. This is done to enhance an individual's accountability for contributing to the teamwork, while also using individual efforts to enhance the quality of the team projects and the efficacy of the overall simulation.

Table 1. Course Units for ITEC 1220: The Politics and Economics of IT

Week 1	Introduction
Week 2	Structuring Cyberspace I
Week 3	The Digital Divide and Social Advocacy
Week 4	Community Networks, Jobs, & Skills in the New Economy
Week 5	User Centered Design
Week 6	Health & Society
Week 7	Extending Cultural & Global Diversity
Week 8	The Environment
Week 9	Software, Monopolies & Intellectual Property
Week 10	The New Economy and Economic Globalization
Week 11	Work and the New Economy
Week 12	Media & Democracy
Week 13	Privacy, Free Speech & Civil Society
Week 14	Structuring Cyberspace II
Week 15	Closing Session and Final Presentations

The entrepreneurial simulation proceeds with the IT Futures Foundation issuing a series of “requests for proposals” (RFPs) that are tied to a specific week’s topic. Thus, during the unit on the digital divide, we might ask the teams to submit a proposal for a community technology center that makes use of the federal E-Rate program. Other RFPs have asked students to address a specific concern having to do with electronic privacy, religious tolerance, and push-button politics. I have also asked students to explore the opportunities for using the Internet to promote small and medium-sized businesses, and to challenge the software monopoly of Microsoft. In each case, the RFPs are written to provide the students with a sense of the underlying issue, a concrete problem amenable to IT-based solutions, and an initial lead on some relevant sources that may help them develop a response to the RFP. We also provide the students with carefully written guidelines about the expected format of the proposal, and a clear statement of the criteria on which their proposals will be evaluated.

The evaluation itself is carried out by means of anonymous review by two of the instructors for the course. In addition to classifying the proposals into categories of special merit (worth 4 points), general merit (2 points), and no credit, the reviewers provide subjective comments in the manner of an NSF proposal evaluation. The top two proposals that are awarded special merit are then selected for formal presentation before the IT Futures Foundation—which is to say, the instructors of the course. This presentation takes place in plenary, in front of all of the students. This helps to cultivate a student’s presentation skills; insofar as the students in the audience see themselves as having to conduct such a presentation in the future, it enhances learning with regard to both content and presentation skills. Teams are awarded additional points for the presentation. The point structure is set to foster healthy competition, and to produce sufficiently strong student works to provide a suitable basis for classroom discussion. A question-and-answer period following each presentation also provides a specific opportunity for instructors to engage with the students, to demonstrate the value of sophisticated analyses, and to establish their credibility in this respect.

Figure 5. Public Debate Session Held in Plenary



At the start of the semester, the IT Futures Foundation also issues a call for a series of public debates on current issues of IT and society. These debates take place on alternate weeks opposite the RFPs, with each team being asked to participate in one debate. The point structure is set so that the debates can account for as much as thirty percent of a team's grade, so that teams put forth considerable effort in researching their assigned topic. We provide a set of relevant questions for each debate topic, as well as a guideline on debating strategies. (This was developed by one of our teaching assistants with prior coaching experience in collegiate debates.) Instructors allow their teams to carry out a mock debate during their class meeting prior to the debate. The debates are evaluated through a combination of the instructor's "vote" (0 to 6 points awarded to each side), a popular vote (3 points split between the two teams based on the students' votes), and a preparation score (0 to 4 points) issued by the team's section instructor. This grading structure was chosen to diffuse some of the competitive aspects of the assignment and to ensure that sufficient points were awarded for research and effort.

The format of the proposals submitted in response to the RFPs, which also applies to the final project, is carefully designed to encourage extensive social analysis. Specifically, we provide the students with a set of guidelines for proposal writing that specified the following format:

- Abstract
- Background
- Problem Definition
- Proposed Solution

- Technical Description
- Implementation Plan
- Budget & Justification

This format requires students to separate the problem definition from their proposed solution, and the background statement from their problem definition. As specified in the course manual, the background statement must therefore describe the general problem of which their problem definition is a specific instance. In the problem definition, students must think in terms of a concrete place and the specific problems faced by stakeholders as they apply to that locale. By requiring the students to specify the problem before presenting their solution, we also discourage students from starting with the solution—a common problem with technically minded students. This point is also reinforced by separating the technical description from the proposed solution. Students are required to provide a general and accessible account of their proposed solution without resorting to technical jargon, or by inappropriately using illustrations and mock-ups to convey what should be articulated within the text. For the RFPs, the technical description is in fact optional. Meanwhile, requiring students to produce a separate implementation plan forces them to think concretely about how they would actually execute their plans. This kind of separation is essential for encouraging students to be more precise in their social analyses.

The final project is a more substantial version of the proposals that the students submit in response to the RFPs. The familiarity of the assignment ensures that the students are able to proceed directly to the substance of their proposal. The instructors are also able to expect greater depth in the students' analyses. Each team is required to identify a possible project and then conduct a "stakeholder analysis," during which they must identify and describe the interests of the different groups that might be affected by their project. This exercise is also timed to coincide with the unit on user-centered design. The stakeholders that they identify also become the chosen subjects for the interviews. All this is submitted in the form of a preliminary proposal, which gives the instructor an opportunity to review the appropriateness and feasibility of a project. The proposal itself is then submitted in three drafts, based on principles about revisions similar to those applied to the IT Rev final project.

The course incorporates a rather wide array of pedagogic strategies, not all of which can be described in the space available in this paper. However, it is worth pointing to some of the dominant strategies. The most important of these is the general strategy of employing educational simulation. As is well documented in the literature on this subject, educational simulation packages a wide array of valuable pedagogic tools and strategies. In general, educational simulations provide an opportunity for students to think through the pedagogic content of a course in very practical terms. Especially in simulations that draw on scenarios that map directly onto student interests, there tends to be a very high level of student engagement. By drawing on real-world examples for the major assignments, students are able to draw on all kinds of tacit knowledge that they already hold about the assignment. Because the student and instructor implicitly know a lot about what the end product ought to "look like," educational simulation also makes it easier for an instructor to recommend substantial changes in a way that is coherent to the student. In addition, the competitive aspect of an entrepreneurial simulation contributes to student work and engagement.

Not only the students benefit from the educational simulation. I have found the work of developing this course to be fairly "intuitive" and easy. The idea of having the instructors serve as an executive board, or of using anonymous reviewers in the evaluation of proposals, all emerged as intuitive steps for supporting the integrity of the simulation. At the same time, these choices often served important pedagogic goals, such as enabling blunt criticism while

displacing the hostility that might be directed to a specific reviewer, precisely because the institution of anonymous reviews were developed in the real world to serve exactly such a function. Moreover, the more we drew on real-world models to build up the simulation, the more the students seemed to accept that careful social analysis was required to produce a good “product.” We have carefully developed many aspects of the simulation, including the official letterhead that we use for all correspondence issued from the IT Futures Foundation. To the extent that is appropriate, we have also used the voice of a professional philanthropic organization in such communications, while reserving our voice as academic instructors for moments where it makes sense to interject a different voice, and to elevate the class conversations to a more reflexive level.

Equally important, we have structured the course to reinforce the authority of the section instructors. Given the competitive nature of the team exercises, the section instructor is quickly seen to be a valuable asset. The critiques offered by the section instructors are rarely viewed negatively, insofar as they are seen as a means of meeting an externally defined standard. The CEOs also serve as an important intermediary between the instructor and other students in the class. The CEOs are usually highly self-motivated students. Moreover, they are acutely aware of the value of the section instructor’s input, so that they help set a deferential tone within the classroom. The CEOs also help create an opening through which the section instructor can reach out to nearly all of the students in their class.

In this course, we also make considerable use of repetition to meet our pedagogic goal of re-socializing our students. For instance, we introduce the students to the concept of user-centered design early on in the semester, and then write this in as a formal requirement for several of the subsequent RFPs. As students repeat the same basic exercise, they come to regard user-centered design not as an abstract concept, but as something reducible to a practical repertoire of design strategies. Likewise, with the weekly writing assignment, the RFPs, and the debates, the students are given an opportunity to internalize a wider variety of concepts and practices into their practical repertoire as IT professionals. Meanwhile, the topical organization of the course prevents the students from coming to view the repetition as a stale exercise.

Repetition, especially when paired with the competitive aspects of the simulation, also plays an important part in encouraging deeper analyses. Over the four years that we have offered the course under the current format, we have always seen the top RFP proposals grow stronger through the course of the semester. More important from a pedagogic standpoint, the teams that struggle with the assignment early in the semester generally learn to mimic the work of the strongest teams. Their improvement is aided by the presentation of the top proposals in class; the written proposals from the top teams are also posted on an online student work archive. Separately, the regularity of the debates, the multiple drafts of the final project, and the demand for weekly writing, all push students to advance their analytical skills.

The course also adopts a novel, and what we regard to be an effective, approach to teamwork. The latest wave of engineering education reforms have brought with it considerable emphasis on the use of teamwork within the classroom. In many cases, teamwork has been implemented poorly, so as to become a detriment to the student’s education. Typical problems include grading structures that encourage freeloading; the lack of assigned roles and leadership within the team; and instructors who are unaware of the common signs and diagnoses for teams that run into serious problems.

In The Politics and Economics of IT, our overall strategy has been to employ very large teams and to encourage internal differentiation in the roles and work performed by the students. By designating a “CEO” and by drawing on implicit understandings about the roles and responsibilities associated with such a position, we have been able to ensure that a clear pattern of leadership emerges within most of the teams. Drawing again on the benefits of a real-world simulation, the CEOs are given the responsibility of distributing the work evenly, and are encouraged to think of how they want to organize their “firm” in a way consistent with the course requirements. CEOs are also instructed to give each student at least three opportunities to contribute to the team, after which the responsibility of convincing the team that they have pulled their weight falls on the individual student.

The grading structure is also designed to facilitate effective teamwork. The overall approach is that of promoting teamwork, but supporting individual evaluation. While teams are assigned a team grade, individual grades may be modified by as much as +1 and -3 letter grades based on an evaluation of the individual’s contributions to the team. Given the opacity of much of the work that takes place within a team, it has been absolutely necessary to develop an instrument capable of revealing the team dynamics to the instructor. We do this through a set of “self-evaluation” forms that the students submit at the end of the semester. These are actually not self-evaluation forms, but evaluations of the team. Students are given the opportunity to state who they felt contributed most to the team and to describe their own contributions. The emphasis is on positive evaluation so students are not specifically asked to complain about specific team members. Nevertheless, the collective input, especially when paired against a separate, confidential evaluation submitted by the CEOs regarding all of the students on their team, provides a fairly coherent view of what transpired within the team. All evaluations are submitted after all work has been turned in, so as to prevent the evaluations from adversely affecting the team dynamics.

Meanwhile, we have found in all instances that certain students step forward to prepare for the debate and others to work on the RFPs. Others start the work on the final project, while others join the effort as the semester draws to a close. The assignments have also been carefully structured so that there is always some component of an assignment—collecting a list of resources for the final project or performing an interview—that can be assigned to specific individuals. This ensures, in most cases, that all students who are willing to do the work will find an opportunity to contribute to the team.

Role differentiation and specialization do not permit all students to develop all of the skills associated with the course. However, our opinion is that specialization is an integral part of effective teamwork, both within academic and non-academic settings. Moreover, as long as professional development is an important objective for this course, it is entirely appropriate that we develop student skills in such a direction. Through effective teamwork, students are exposed to the skills and strategies of other students. Given that our students are still within the formative stages for defining their personal and professional identities, the course provides students with many role models and imagined identities that they may be willing to experiment with in the future.

Clearly “Politics and Economics of IT” does not speak directly to the theme of using history to teach computer science. However, the course delivers the same kind of humanistic content that can be found in a course on the history of computing. Paired with “IT Revolution, Myth or Reality?” it also provides a set of pedagogic exercises that drives home many of the lessons and techniques of social analysis delivered in that course. More generally, the pedagogic experimentation in “Politics and Economics,” and the use of entrepreneurial

simulation, may be something that can be transferred into a course rooted more firmly in the history of computing, especially where the curriculum will not permit a two-course sequence on the general topic.

Conclusion: Experiences and Assessment

We have been very pleased with the innovative humanistic curriculum we have been able to develop for the IT Program at Rensselaer. However, we do not wish to suggest that these two courses present no challenges. Insofar as an honest assessment of our experience will prove most useful for anyone who wishes to contemplate a similar undertaking, we would like to offer an open discussion of our experiences, and the accomplishments and limitations of our courses. The following comments are not based on any formal assessments, although we have relied fairly extensively on student input and have scrutinized the standard assessment forms issued by our institution.

With regard to "IT Rev," the greatest challenge we face continues to be that of convincing the students that humanistic content is relevant to their interests. History of computing, when offered as an elective, tends to be popular with computer science students. Yet, when offered as a required course, a significant fraction of the students fail to see the relevance of the course, and fall back on the technical interests that led them to choose their field of study. Lack of engagement has a direct effect on the willingness and ability of these students to adjust to the workload. Various strategies for curtailing the amount of reading that they do can appear.

It is important to emphasize that not all students are affected in this manner. Many accept the underlying rationale for the course. Most do a substantial portion of the assigned readings. Some of the students who struggle with the reading and writing assignments early on in the semester develop the skills that we aimed to cultivate, and find that they are able to accept the workload by mid-semester. Our students also appreciate the fact the course draws on popular readings, and that they are allowed to continue viewing information technologies in revolutionary terms, even as they are given the social outlook necessary to assess what constitutes a radical change.

All the same, the substantial resistance that we encounter, especially in the first course, points to the considerable challenge of reorienting students away from an exclusively technical orientation. It has been essential for the instructors to use an array of pedagogic strategies for engaged learning: the use of brief video clips, a particular plenary-recitation format, activities where students are allowed to *do* things rather than simply reading and discussing texts. Where there are implicit tensions with the instructor, breaking up the class into small group discussions, and allowing the students to discover the relevance of the readings through peer-to-peer learning strategies, has proved to be quite valuable. It is also possible that the structure and content of the course is such that the section instructors find it easier to distance themselves, in the eyes of their students, from the lead instructor. This can have detrimental effects for the lead instructor's teaching evaluations.

Meanwhile, the gravest challenge for The Politics and Economics of IT has been the dot-com bust. Having built the program on a student cohort that was exceptionally attuned to career prospects and market opportunities, the collapse of the IT job market has led to a general flight away from IT as a discipline. First-year student enrollment within the IT program has dropped precipitously from a peak of more than 120 students in the entering class to just over 30 last year, and a projected enrollment of only 19 first-year students for the 2004-05 academic year.

These numbers grow significantly through students who transfer into the program. Nevertheless, the large-scale entrepreneurial simulation depends on having at least 70 to 80 students. Below this number, the course structure introduces strains in terms of workload, and the lack of sufficient effects of competition as far as bringing student work up to a higher standard.

Apart from this issue of enrollment, it is also not clear how effectively the benefits of the course are distributed among the students. Practically all of the “CEOs” emerge from the course quite excited about the experience. Nevertheless, by placing so much emphasis on teamwork, and on simulation in general, the course creates a scenario where certain individuals may drop back from the course and deliver satisfactory performance without ever really engaging with the course’s pedagogic intent. The section instructors may themselves be distracted by the team exercises, and fail to pay attention to individual students who do not fully engage with the course. The course currently earns a fairly high evaluation; nevertheless, the distribution of scores suggests that there is a substantial body of students who do not find the course as effective as some of their peers. It may be important to think about how to advise the section instructors to work with this particular challenge.

At a more mechanical level, we have also had to contend with perennial complaints about the complex grading structure for the course. In some sense, this is endemic to any large-scale entrepreneurial simulation. In order to make a large-scale simulation work, it becomes necessary to develop a diverse array of assignments that makes the simulation realistic. Moreover, given that we encourage students to specialize in their contributions to teamwork, this requires a grading structure that supports all students’ choices about what contributions they make to their team.

There are a number of other things worth mentioning. Drawing on contemporary and popular readings, especially involving URLs that frequently change, does pose a substantial burden in terms of maintaining the course website and its readings list. Entrepreneurial simulation also places substantial demands on course management. This is not an aspect of teaching that will be attractive to all faculty members. Teamwork also tends to collapse and collapse unexpectedly. Again, it is essential that instructors be trained to recognize the common ways in which teams run into difficulties.

Lastly, there are times when we have had to ask whether the Faustian bargain applied more to us, as instructors, than to our students. Given that the course draws simultaneously on the notion of entrepreneurialism and social advocacy, it is not always clear which aspect of the course students walk away with. In general, they are more excited by the fact that they were able to participate in a major entrepreneurial exercise. While the CEOs speak about “having learned much from the course,” they point to what they learned about entrepreneurialism and management first, and to social and economic analysis as only secondary to those skills. On the other hand, to the extent to which the social and economic aspects of doing IT work has become integral to their definition of management and entrepreneurship, the course remains on track with regards to its stated objectives. The students also suggest that both courses have introduced them to many social and historical aspects of information technology that they had not thought about before. This is certainly encouraging from the standpoint of liberal education.

What has pleased us most, however, is the tremendous sense of accomplishment that many of the students have derived from both courses. Writing a thirty-page paper may have seemed a daunting task to students who had never written a college paper. Yet, by the end of IT Rev, all of the students who took the assignment seriously could point to the substantial

research paper that they produced. They also emerge from the exercise with a sense of having developed a much broader appreciation for the social dimensions of a technology. Meanwhile, students emerge from The Politics and Economics of IT with a sense that they know what it means to do IT in the real world. Surely, the course will have introduced its own biases and misperceptions. Yet, we find it significant that the students emerge feeling confident, and empowered, in their ability to function as an IT professional.

Figure 6. Large-Scale Entrepreneurial Simulation and Student Engagement



At a more concrete level, it is probably useful to list some of the skills that we feel these courses confer to our students:

- An ability to “read” the social dimensions of information technology
- An orientation towards multiple causes, as opposed to a linear model of technological change
- Critical reading skills and reading endurance
- Introduction to different modes of communication and dialogue, including
 - Seminar discussion
 - Formal debates
 - Informal essays
 - Position papers
 - Proposal writing
 - Presentation skills
- Stakeholder analysis
- Interview skills
- Teamwork and team-building skills
- Developing a close working relationship with an instructor

In addition, the two courses provide our students with an extensive body of historical and contemporary knowledge about the social dimensions of information technology.

Again, the syllabi for the two courses are presented in Part III of this volume. The course website for The Politics and Economics of IT, including the detailed readings list and an archive of student work, may also be found at the following URL: www.rpi.edu/~akeraa/pe-IT. We welcome all thoughts and inquiries about our two courses.

Atsushi Akera is an Assistant Professor in the Department of Science and Technology Studies at Rensselaer Polytechnic Institute. He is currently working on a book, *Calculating a Natural World: Computers, Scientists and Engineers During the Rise of U.S. Cold War Research*, which uses the history of computing as a lens into the early U.S. institutional infrastructure for cold war research. *Author's address:* Department of Science and Technology Studies, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 USA. Email: akeraa@rpi.edu.

Kim Fortun is the associate dean of the School of Humanities and Social Sciences, and associate professor in the Department of Science and Technology Studies at Rensselaer. She is also the Director of Rensselaer's Center for the Ethics of Complex Systems. She received her doctorate in cultural anthropology at Rice University, and is the author of *Advocacy after Bhopal: Environmentalism, Disaster, New Global Orders* (University of Chicago Press, 2001). Her current work includes the study of information technology, environmental justice, and social stratification. *Author's address:* Department of Science and Technology Studies, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 USA. Email: fortuk@rpi.edu.

7. The Synergy between Mathematical History and Education

Thomas Drucker
University of Wisconsin-Whitewater

Abstract

Mathematics has a long history, and there have been many attempts to build some of the material from that history into courses of mathematics at various levels. Simply including pictures of distinguished mathematicians from the past does not tend to attract or hold the students' attention. If it is possible to give the students a sense of how the discipline has changed thanks to the efforts of individuals, those students can come away with a recognition that they may have the chance to make a difference themselves. They also can tag certain arid aspects of a theoretical discipline with something to associate with the rest of their lives.

Isaac Newton wrote, "If I have seen farther, it is by standing on the shoulders of giants."⁶³ This dictum has been applied for centuries by mathematicians to keep their opinion of their own contributions within limits. It also serves as a guide to the ways in which the history of mathematics can play a useful role in the mathematics classroom. This suggests that the teacher of computer science might find some guidance in bringing history into the classroom as well.⁶⁴

One point that students can hardly miss with even a minimal historical perspective is the extent to which they are able to do their work, thanks to the efforts of many predecessors. The ahistorical perspective of college students is well known, but bringing them face to face with problems solved in the past can remind them of how far they would get without the methods developed earlier. In this way, a student is less likely to feel that what the course (text, instructor, and so forth) has to offer is irrelevant.

In addition, however, the student also has the chance to learn what discovery is like. It is hard to know whether any contemporary advance will permanently change the shape of a discipline. We have plenty of evidence from the past of happy accidents that resulted in previously unsolved problems being tamed. If a student can feel part of the community that solved such problems, there is less danger of paralysis when confronted with something not entirely like what has been seen before.

There is no shortage of material to bring into the classroom regarding the work of the greatest mathematicians (e.g., Gauss, Archimedes, and Newton). On the other hand, even their accomplishments have not always been equally fruitful, and one can point to byways in their work that have not decisively affected their successors. A point to draw is that there is no way of telling in advance just how fruitful any research program is likely to be. Fortunately, at early enough stages in their careers, students do not need to worry so much about issues of funding in the absence of immediate positive results. Examples from the past are usually less entangled with the business of procuring grants.

There aren't many courses into which Fermat's Last Theorem can't be introduced. In geometry it can be treated as a generalization of the Pythagorean Theorem, and in algebra it is

⁶³ For the history of this aphorism, see Robert K. Merton, *On the Shoulders of Giants* (Chicago, 1993).

⁶⁴ As a historian of mathematics, I can only hope that the guidance furnished by the case of mathematics will be useful to computer science. In general, I shall not be drawing explicit morals for computer science, allowing the lessons of mathematics to speak for themselves, *mutatis mutandis*.

just a matter of looking for whole number solutions for a certain kind of equation. While few students are likely to be able to follow the story to its conclusion, the telling illustrates the way in which some advances can prove to be more fruitful than others. It is also a useful illustration of how a relatively concrete problem can sweep those trying to solve it well beyond the realm of the mathematics in circulation at the time of the problem's statement.

Probability offers another area in which a concrete problem leads to far-flung (and sometimes implausible) results. The use of the 'Let's Make a Deal' problem in the classroom encourages students to take an intuitive crack before going ahead to present the machinery that can clarify the correct approach. Describing the response to the correct solution in the popular press can also illustrate the extent to which even those involved in teaching mathematics can be liable to mistaken intuitions.

Another lesson offered by probability is the extent to which one can make progress with calculations even when one is unsure of the foundations upon which one is building. The problem of the foundations of the calculus can be a complicated historical issue, but there is a sense in which the mathematical questions found some resolution in the work of nineteenth-century analysts like Cauchy and Weierstrass. Problems about the foundations of probability can remain vexing, despite the variety of mathematical techniques now being fit into a probability course. The challenges on the interface of mathematics and the 'real world' illustrate the difficulty of fitting any model to a particular situation.

The business of turning a challenge into a positive result can be well illustrated by Gödel's incompleteness theorem. The student can be confronted with 'I lie' and regard it as little more than a parlor trick. Attempts to fill in a few more of the details with 'This statement is false' might not carry any more conviction. The account of how Gödel transformed the paradox into a theorem only gains in impressiveness as the student sees the far-reaching usefulness of Gödel's result. The applicability of Gödel numbering to Turing machines makes this especially appealing in a computer science context.

On a slightly more philosophical level, Brouwer's intuitionism comes with a good deal of psychological underpinning. The idea of being able to transform Brouwer's perspective into a constructive research program should help students to learn that trying to introduce clarity can have side effects as well. The translation may not be faithful to all of Brouwer's philosophical preferences, but computer scientists can enjoy the benefits of constructive formal systems without trying to do full justice to their sources in Brouwer's own thought.

One issue that the conscientious historian cannot avoid, but which may not be all that central to the educator, is the truth of the anecdotes with which one salts one's presentation in the classroom. The classic example is the stories one finds under the heading of biographies in the work of E.T. Bell.⁶⁵ They are memorable stories, but they are often deplorably inaccurate.⁶⁶ Mathematicians continue to say that they were influenced strongly by Bell's writing to go into mathematics. This raises the question of whether somehow bad history is, in this case, making for good pedagogy.

The point is not that good history has to be dull. What one can do in the classroom is to use the stories that Bell (and others who follow his taste for liveliness at the expense of

⁶⁵ Eric Temple Bell's *Men of Mathematics* (originally New York, 1937, but many editions since then) has long been a *bête noire* for historians.

⁶⁶ See, for example, Tony Rothman's essay on Galois in his *Science à la Mode* (Princeton, 1989).

exactitude) offers and to point out one or two of the cracks in the façade. In a mathematics classroom one is not primarily concerned with indicating Bell's shortcomings. It is a good setting, however, for revealing the extent to which easy assumptions can distort one's efforts to describe what happened in the past.

The assumption to which many telling the story of the mathematical past are most liable is the scientific version of the Whig interpretation of history. This attitude (to which we are all occasionally tempted to succumb) tends to find in the work of great mathematicians more of the subsequent development than is demonstrably there. Here it is quite difficult to avoid leading the students in the direction that mathematics has taken, since that is what one is primarily trying to get across. If it is not unfair to educators, perhaps one can claim that a progressivist bias is the occupational hazard of those who seek to bring history into the mathematics classroom.

After all, there is only a limited amount of time, and most syllabi tend to be more in danger of bursting at the seams than offering the opportunity to pursue additional topics. One would like to be able to correct a misimpression on the students' part of the static nature of mathematics, but there are frequently final examinations and prerequisites for other courses for which the students need to be prepared. The justification of the time devoted to historical issues in a course can't be separated from how much time those issues will usurp from mathematics. If an historical note will make it easier for a student to remember the quadratic formula or the binomial theorem, that is bound to seem more valuable than simply clarifying the use of τ in Greek mathematics.

If there is a way of justifying the intrusion of history into the mathematics classroom, then surely it is by virtue of the human face that students can associate with the doing of mathematics. Pedagogy has recognized the value of giving students not problems artificially concocted on the basis of the methods already presented in the text, but situations taken more immediately from the world outside the textbook. It is admitted that dealing with such problems takes a good deal more time than simply being able to plug the numbers into the obviously applicable formula. On the other hand, confronting problems that the world presents is better training for whatever subsequent use the student is likely to make of mathematics.

In a similar vein, history can offer the examples of specific problems that caught the imagination of mathematicians and led to the creation of new fields. The particular game that the Chevalier de Méré was playing may not be the dice game with which students are most familiar, but Fermat's and Pascal's embroidery on the answer to his question helped to get probability back into mathematical focus.⁶⁷ The quest for an alternative to the parallel postulate led to the formulation of non-Euclidean geometries. From specific problems in the past, individual mathematicians have created disciplines upon which aspects of our culture not limited to mathematics depend. Most of today's students will not loom like giants over the generations to come, but history has the chance of reminding some of them that they can give a boost to others by following up the problems of today.

⁶⁷ Anders Hald, *A History of Probability and Statistics and Their Applications before 1750* (New York, 1990).

Acknowledgments

Thanks for comments on my initial ideas for this paper are due to William Aspray, Mehmet Dalkilik, Thomas Haigh, Charles Huff, and Jennifer Light. Technical assistance was furnished by Shelley Matthews. This article would never have been written at all without the guidance going back many years of Michael S. Mahoney.

Thomas Drucker is a lecturer in the Department of Mathematical and Computer Sciences at the University of Wisconsin-Whitewater. He is editor of the *Bulletin* of the Canadian Society for the History and Philosophy of Mathematics, and edited *Perspectives on the History of Mathematical Logic* (Boston, 1991). He has taken part in the discussion about the role of truth in presenting history in the mathematics classroom. *Author's address*: Department of Mathematical and Computer Sciences, University of Wisconsin-Whitewater, Whitewater, WI 53190 USA. Email: druckert@mail.uww.edu.

8. Computing for the Humanities and Social Sciences

Nathan L. Ensmenger
University of Pennsylvania

Abstract

This paper describes a course in computing for students in the humanities and social sciences developed at the University of Pennsylvania. Taught by an historian of computing, it made extensive use of history in the teaching of technical subjects in computer science. A description of specific uses of historical material is provided. The course syllabus may be found in chapter 18 of this volume.

Information science and technology are ubiquitous components of the intellectual and pedagogical life of the modern university. Not only have concepts drawn from computer science and information theory directly affected the content of numerous disciplines in the sciences and humanities, but the widespread use of computer technology in the classroom, laboratory, dormitory, and administrative offices has fundamentally altered the ways in which students and faculty explore and practice their respective disciplines. Understanding what computers are, how they work, and how they can best be used for research and presentation is an essential element of a well-rounded undergraduate education in any discipline.

How can such an understanding best be developed and communicated, however? The challenges associated with teaching general-audience science courses are well documented but difficult to overcome: such courses (often given such whimsical titles as “Physics for Poets” or “Rocks for Jocks”) run the risk of trivializing the discipline or presenting a misleading illusion of simplicity or mastery. Instructors cannot assume that non-majors possess the basic skills that are fundamental to the discipline (advanced mathematics in the case of physics, for example), and, therefore, need to provide either remedial training in such skills (spending valuable course time in the process) or attempt to proceed without them (thereby providing an impoverished perspective on the discipline). This is a particular problem for introductory courses dealing with computing: not only do students have to come to grips with the scientific content of the course, but also with the technology required to put this content into practice. Courses intended to teach broad “concepts in computer science” must often necessarily also provide “basic skills in computing” training. The conceptual and logistical problems posed by such a combination are formidable.

Several years ago, the School of Arts & Science at the University of Pennsylvania developed a course aimed at providing its students with an introduction to computer and information science. The overall focus was to be on the fundamental concepts that transcend any particular technology; the goal was to provide students with the knowledge needed to readily and skillfully adapt to a rapidly changing technological environment. As an added benefit, we expected that in doing so they would also acquire skills that will be relevant to them in other coursework, and possibly even their future careers. Although the course was not specifically intended as an alternative to a traditional introduction to computer science, in practice it ended up serving as just such an alternative, at least for a certain population of undergraduates. Because the course was taught (for various intellectual and institutional reasons) by an historian of computing, it provides an excellent example of how history can be incorporated into the computing curriculum. What follows is a description with our experience of the difficulties (and rewards) inherent in developing and implementing such a course; hopefully it will prove useful to readers interested in developing historically minded computer science courses at their own colleges and universities.

First, a note on the specific institutional context of this particular experiment is needed. At the University of Pennsylvania, the computer science department is located in the School of Engineering and Applied Science, rather than the School of Arts and Sciences as is typical in many American universities. This makes it particularly difficult at an administrative level to coordinate curriculum developments in the humanities and social sciences with those in computer science. In the late 1990s computer science was experiencing its own difficulties accommodating the surge in enrollments brought about by the dot-com explosion; they had no resources left over to staff curricular experiments aimed primarily at students in the School of Arts and Sciences. And so the School of Arts and Sciences decided to develop its own alternative introductory computing course. From an historical perspective, its decision to do so was in part practical (it was felt that an historian of computing would be one of the few humanities or social science specialists capable of providing such a course), but primarily pedagogical: history was seen as a useful tool for teaching this material.

Developing the Course Objectives

There are several questions that need to be addressed during the development of any new introductory course. Who is the intended audience, and what would we like them to accomplish? Will the course be designed to provide a broad survey of the discipline, or to develop specific skills that are prerequisites for more advanced study? What topics will it cover, and at what level of detail? What background, skills, and abilities will be required of both the students and the instructor?

In our case we decided that we wanted to target students in the humanities and social sciences who: a) would benefit from exposure to the concepts and perspective provided by computer and information science (namely all of them); and b) were otherwise unlikely to take courses offered by the computer science department—either because they were limited in enrollment, were perceived to be too difficult for non-majors, or did not fulfill particular School of Arts and Science requirements. Our goal was to provide these students with a basic understanding of what computer science is and how it is relevant to a broad range of intellectual and occupational disciplines. We also wanted them to explore ways in which the humanities and social sciences—history, economics, political science, sociology, and anthropology—could, in turn, contribute to our understanding of the broader social implications of computer science. We also hoped that our course would provide some practical tools for their research in their own home disciplines, and might even serve as a “gateway” into more advanced courses in the computer science department. Here is our tentative description of our agenda as developed very early on in the development process:

The goal of this course is to provide a broad overview of fundamental concepts. We want the students to learn enough specific information to make meaningful assessments and to get useful work done; we do not want them to spend too much time learning the trivial details of the latest programming language-of-the-week. We also want to focus on more than techniques and mechanics: the goal is to provide students with a broad perspective on information technology, one that will allow them to critically evaluate the many claims that are made about the power of the computer to change lives and drive social developments ... The best way to accomplish both of these goals is to closely integrate the social and technological components of the course content. In our section on computer programming, for example, we cover some of the fundamental technical concepts of programming, as well as some of its more interesting sociological and anthropological dimensions: the formation of a unique programming subculture and “hacker ethic”; the aesthetics of programming and its relationship to such traditional arts as literature and mathematics; the political libertarianism of the “open source” software movement.

As is evident from the latter paragraph, history was to play a central role in the accomplishment of this agenda.

The Course Structure

The complete course syllabus is provided in chapter 18 of this volume. What follows is a brief description of the ways in which history was used to complement the technical material of the course.

Unit I: Computers and How They Work

One of the goals of the course was to de-mystify the process of technological development. We wanted students to have confidence in approaching the topic of computing and to understand that although computers are complex, they are also comprehensible (even to those without a highly technical background). Unfortunately, because the heart of the modern computer is quite literally a “black box” (or at least a “black chip”), it is a particularly difficult technology to deconstruct (both literally and figuratively). Even students with training in computer science or electrical engineering might only have encountered the inner workings of a computer in terms of abstract representations of idealized architectures. By looking at the electronic computer as an evolutionary rather than revolutionary development (that is, through the history of its long social and technological development), its internal structures (and the technical, social, and economic compromises embedded therein) are particularly apparent.

For example, in our first week we showed that the electronic digital computer did not just appear out of thin air in the mid-1940s. It is part of a whole series of social, technological, and economic developments dating back to at least the early 19th century. We explored the multiple origins of the modern computer as business machine, communications device, and scientific instrument. In this we had at least two agendas: 1) to situate the electronic computer in a long trajectory of technologies aimed at storing, retrieving, and manipulating data; and 2) to show that ideas about what a computer was and what it could be used for have changed over time. We used the history of scientific computing (orreries, tide predictors, and the differential analyzer) to explain the difference between analog and digital approaches to computing.⁶⁸

After briefly exploring the history of the ENIAC (an example particularly relevant as it was developed at the University of Pennsylvania), students learned (in a single lecture!) how to construct an electronic computer out of a battery, switches, and light bulbs. And not just any computer, but the most powerful computer that has ever existed or ever will! The Alan Biermann book, *Great Ideas in Computer Science*, was indispensable for this purpose.⁶⁹ If carefully planned in advance, this lecture can be accomplished in just one forty-five-minute session. This was probably the most difficult technical material of the semester, but keep in mind that the goal is not to have the students design their own counting circuits, but rather to provide them with a

⁶⁸ *A note on pedagogy:* This historical introduction to the computer also provided an excellent opportunity to talk generally about the student’s perceptions of the computer: what they are, what they are good for, who uses them, what is good and bad about them, etc. This discussion set up some conceptual and technical questions that could then be addressed later in the semester.

⁶⁹ Biermann, Alan. *Great Ideas in Computer Science: A Gentle Introduction*, Second edition (Cambridge, MA, 1997), chapter 7, “Electric Circuits.” Note that this chapter was eliminated from subsequent versions of the Biermann text. For the purposes of this course, the 1997 edition is essential.

basic sense of how a digital computer works (as well as the confidence that an even deeper understanding is not necessarily beyond their grasp).

From the specific we now moved to the general. The history of the ENIAC and our own experiment with simple digital circuits illustrated the techniques and challenges associated with engineering a computer; from there we moved on to the science of computing. Again, a relevant historical case study (John von Neumann's *First Draft of a Report on the EDVAC*⁷⁰) provided a relatively accessible account of one of the central concepts in computer architecture.

Since one of the more significant developments that occurred in the early 1950s was the transformation of the computer from a purely scientific and military instrument into a tool for corporate control and communication, we spent some time discussing just how and when this occurred. This transformation influenced not only the character of the technology, but also the shape of computer science as an intellectual and occupational discipline. The technical material for this week focused on the development of real-time computing technologies and the history of the transistor—both key technological and conceptual developments in computing—as well as their respective influences on the structure of computing (and the computer industry).

Unit II: Introduction to Programming

At the heart of modern information technology is the software that controls it. In this section we attempted to provide students with a basic understanding of what it means to program a computer. This four-week section began with an introduction to the programming language Java. Again, the real goal was to get at the fundamental concepts that transcend any particular implementation of a given technology. In the first two weeks we covered the big concepts: high-level languages vs. machine code; compiled vs. interpreted; functions and procedures.⁷¹

We began our discussion of computer programming by looking at the history of programming, which led us naturally from our earlier study of hardware architecture into a discussion of machine languages. Chapter 8 (“Machine Architectures”) of Alan Biermann's *Great Ideas in Computer Science* provides an excellent introduction to machine-language programming. It works well even for the least technically inclined students. It also makes quite clear why the development of higher-level languages was so important to the widespread diffusion of the electronic computer.

As we moved quickly into a discussion of conditional branching and looping structures, we also briefly explored some of the non-technical aspects of programming: style, elegance, “the Art of Programming.” The goal was to have students learn to appreciate programming as

⁷⁰ John von Neumann, “First Draft of a Report on the EDVAC,” Contract No. W-670-ORD-492, 30 June 1945 (Philadelphia: Moore School of Electrical Engineering, University of Pennsylvania). Reprinted (in part) in Brian Randell, *Origins of Digital Computers: Selected Papers* (Berlin, 1982), 383-392.

⁷¹ *A note on programming languages:* Although we were not particularly wedded to any particular programming language, we wanted to use a language that was easily accessible, freely available on a wide range of platforms, and that could actually be used to get some work done. In our first attempt at teaching this course at the University of Pennsylvania, we felt pressured to use Java—in part because at the time, knowledge of Java was seen as a useful occupational skill, and in part because it was then being used in upper-level computer science courses. Although Java is an excellent programming language in many respects, it proved needlessly complex for our purposes—far too much overhead (in terms of initialization and the loading of libraries) was required for even the most basic programs. In retrospect, it might have been better to have used a scripting language such as Perl or Python, which is both easy to learn and useful for the kind of activities (text manipulation, web-content creation, etc.) in which humanities undergraduates are likely to engage.

both an art and a science akin to both mathematics and literature. Obviously all of these issues could be addressed only at the most superficial level. The focus was (as much as possible) on high-level concepts rather than syntax—another argument for using a simple scripting language.

One of the most exciting and interesting aspects of computer technology involves the simulation of real-world environments. The creation of such computer models—economic, biological, and social—has dramatically altered the way we understand the world, formulate public policy, produce scientific knowledge, and think about our minds, bodies, and culture. After the students had learned enough basic programming to put together a basic program, we worked together to develop a simple simulation program that tracked the progress of a disease epidemic. In doing so we introduced several new programming concepts: input-output, file handling, and string manipulation. More importantly, however, we were able to get the students excited about the research possibilities enabled by the computer. Although our simulation was relatively simple to implement even with the rudimentary programming skills the students had learned thus far, it was complex enough in its behavior to make them feel as if they were doing something interesting. It also generally provoked a fascinating discussion on the epistemological/ontological issues associated with this particular use of the computer.

Many introductory programming courses present only “toy” problems and elegant but trivial examples. This often leads casual or non-programmers to underestimate the complexities of real-world programming applications. Our last week on programming was devoted to introducing students to the many tasks required to produce a meaningful solution to an actual research problem: analysis, iterative design, debugging, and optimization. The objective was not to make them expert software developers, but rather to give them an appreciation for how programming is done, with an eye towards a more critical evaluation of how computers can best be applied to relevant real-world problems.

Unit 111: Computers and Society

In this section, we explored the rise of the personal computer and the Internet as both technical and social phenomena. This allowed us to discuss the development and structure of microcomputers and computer networks and some key underlying technologies, including information theory, packet switching, and protocols. The focus, however, was on the relationship between technical developments in computer science and theories about the meanings and origins of the “information society.”

After an historical introduction to the ARPAnet and Internet, which provided a useful perspective on the nature and significance of communications protocols, we walked the students through the basic concepts involved in Web content creation and presentation, most specifically structural markup languages (HTML). We also addressed some larger conceptual and design issues. For example, presenting information effectively on the web involves more than just displaying an electronic version of a paper document. The Web has its own conventions, styles, strengths, and limitations. Learning how to read a web-based “text” is an important research skill for students in every discipline, and involves skills that are not necessarily acquired as part of a traditional humanities education.

Continuing our discussion of the changing social meaning of information technology, we turned our attention to the computer as a tool for communication. We also explored the ways in which information technology is adopted by contemporary social movements: utopians, social and political revolutionaries (and reactionaries), patient advocacy groups, and others. In terms

of technology, we talked about compression and encryption algorithms and their relationship to privacy and intellectual property issues.

Conclusion

Ours was obviously a course that attempted to meet many (probably too many) objectives: it was equal parts introduction to computer science, basic computer literacy, and history of computing. There was clearly not sufficient time in a single semester to satisfactorily accomplish all of these objectives. Nevertheless, the course was generally considered to have been successful, both by the students and the instructor. What is most interesting for the purposes of this paper is how useful the historical materials turned out to be. It was much easier to teach basic computer architecture in terms of specific historical examples than as idealized abstractions. Students were better able to visualize the inner workings of the computer when they were described in terms of more familiar concepts—electrical circuits, arithmetic operations, and counting devices. In a similar manner, the historical development of programming languages—the shift from machine language to compiled languages—provided the students with both a means and incentive to understand how software really works. Lastly, many of them felt that it was the bigger picture—the larger social and social economic context of computing—that was most relevant and useful to them in their own work and careers. If anything, they wanted to learn more history, not less.

Nathan Ensmenger is an assistant professor in the Department of the History and Sociology of Science at the University of Pennsylvania. His current research focuses on the intersection of work, computers, and organizations. *Author's address:* 303 Logan Hall, University of Pennsylvania, 249 S. 36th Street, Philadelphia, PA 19104 USA.
Email: nathanen@sas.upenn.edu.

III. Specific Courses and Syllabi

Course Descriptions & Syllabi (chapters 9-13)

9. History in the First Course in Computer Science

J.A.N. Lee
Virginia Tech

Abstract

Using history as a supportive component of the teaching and learning of topics within the first course in computer science can provide an extra dimension to the memorization of the subject, while at the same time contributing a humanistic view of a highly technical field. While the majority of today's students were born after the advent of the personal computer and have never known a world in which a company like Microsoft did not exist, most are totally ignorant of the stages of development that led from the first machines to the Information Age in which they live. The faculty of our teaching institutions have a responsibility to provide our students with the best information about the field in which they desire to be employed, and the means by which we got there. This paper describes one such attempt, and discusses the opportunities and challenges for a broader infusion of history into the computer science curriculum as we prepare to modify our programs to conform to "Curriculum 2001."

In any curriculum, the introductory course is the key to maintaining the students' interest and enthusiasm and encouraging them to persevere to comprehend the concepts and fundamentals and, where appropriate, to develop skills that will serve them well in the rest of the curriculum. Computer science has had two "traditional" introductory courses—a "computer literacy" course and a beginning programming course. In terms of "Curriculum 1991,"⁷² these are generally designated as courses CS0 and CS1. While the contents of the literacy course have been assumed to be in the high school preparatory repertoire in many curricula, it is an essential element of the program of study, laying the groundwork for many later courses. It also demonstrates to the participants that there is more to the field of computing than just programming. CS1 has the advantage of being a "hands-on" course, holding the curiosity of the students and allowing a measure of exploration that is important to their continued pedagogical development.

The major purpose of the project described here was to bring some laboratory-style activities to the literacy course that would retain the interest of the participants, providing an exploratory zone built into a web-based system that would be capable of being used in a distance-learning application. It was recognized early in the development cycle that this course could be built around the demonstration of abstract historical artifacts and that the historical backdrop of each would provide a memory-catching legacy to reinforce the learning activity.

The approach taken was to develop a learning plan that is primarily a "grand tour" of computer science that can easily be tailored to the curriculum of most any program. Like a geographical grand tour, the introduction to computer science should explore the main avenues of the subject, with enough of a view of the alleys and byways to suggest to the traveler that

⁷² Allen B. Tucker, Bruce H. Barnes, Robert M. Aiken, Keith Barker, Kim B. Bruce, J. Thomas Cain, Susan E. Conry, Gerald L. Engel, Richard G. Epstein, Doris K. Lidtke, Michael C. Mulder, Jean B. Rogers, Eugene H. Spafford, and A. Joe Turner, *Computing Curricula '91* (New York, 1991).

they can come back later and explore on their own, or pick up another tour guide (in another course) to look at the deeper details. Just as a bus tour of Washington, DC, shows the tourist where to find the primary tourist sites such as the Lincoln Memorial or the Smithsonian Institution, so this course shows the locations and facades of the key topics within computer science. Each learning module is set up to contain three major elements (see Figure 1)—pre-class activities to be completed by the student, in-class activities to be managed by the teacher, and post-class projects that reinforce the learning experience. In the distance learning version these are modified to become the pre-study, study, and post-study activities. Besides written descriptive materials, each segment contains laboratory style projects and demonstrations.

Three versions of the course are currently being offered:

1. A traditional lecture and homework version (sage on the stage);
2. A web-supported version (guide on the side) with help sessions, on-line laboratories and assignments; and
3. A distance-learning version (cox in the box).

Figure 1. Lesson Outline



Topics Covered

The Virginia Tech version of this course covers the following topics:


- What is Computer Science (or not?)
- Problem Solving
- Algorithms
- Efficiency of Algorithms
- Building Blocks of Computer Systems
- Number Systems and Arithmetic
- Boolean Algebra and Logic
- Gates and Circuits
- Computer Organization

- Virtual Machines
- Software Engineering
- Programming Language Paradigms
- Compilers
- Models of Computation
- Applications and Networks
- Social Aspects of Computing

Curriculum 2001 suggests an introductory course (see Appendix A) that covers the same topics although with slightly different titles, with the exception of the last two items that are omitted altogether. Each of these topics has a collection of demonstrations and projects associated with them and, where appropriate, an historical vignette (see Figure 2) to provide a background. Within each topic there is a common outline:

- Basic definitions
- Historical origins
- Subsequent course links
- Further development of the topic starting from the basics

Figure 2. Historical Vignette Accompanying Unit on Algorithms

	<p><i>The whole course is based on the study of algorithms and thus we start from a study of the origins of algorithms—and thus we start with an introduction to Mukhammad ibn Musa Al-Khowarizmi [Zemanek, 1979] the inventor of the algorithm and from whose name the term is derived. In approximately 825 AD, Al-Khowarizmi wrote a treatise on equations. The title of this Arabic work was <i>Hisāb al-jabr w'al-muqābala</i>, meaning "The calculation of reduction and restoration." This meant the transposing of negative terms to the other side of the equation (restoration) and uniting similar terms (reduction). The "al-jabr" was translated into Latin by Robert of Chester in 1140 AD. This Latin translation, <i>Liber algebrae et almucabala</i>, popularized the name algebra for the theory of equations. Another arithmetic of Al-Khowarizmi was translated into <i>Liber Algorismi de numero Indorum</i> (the book of Al-Zemanek.Khowarizmi) giving rise to the word algorithm. (http://courses.cs.vt.edu/~cs1104/Algorithms/AIKhowarizmi.htm)</i></p>
<p>Photo by Heinz Zemanek</p>	

Throughout the course this approach to incorporating history is followed—that is, associating the origin of a concept or machine with an “inventor.” In some cases this raises difficulties, such as in the choice of the inventor of the computer (the most frequently asked question on my history web site <http://ei.cs.vt.edu/~history>). In many cases, the fact that the object in their hands (literally) did not just appear but had a true inventor is a great surprise to the student.

Overcoming Myths

In the study of computer systems we look at the history of the development of the computer and the five “inventors” of the computer (Konrad Zuse, John Atanasoff, Howard Aiken, George Stibitz, and Alan Turing), and the fact that in the 1930s the “primordial stew” of computational technology contained all the necessary pieces to enable them to assemble the first machines. This overcomes the myth promulgated in most textbooks that the first computer was the ENIAC. In fact, it has been found to be necessary to provide “caveat emptor” statements to the readers about such myths in their textbook. Similarly, the origin of the term “von Neumann architecture” is used to discuss the description of the EDVAC, which does not have an architecture similar to that ascribed to the von Neumann machine, and the “Draft Report” that included the name of only one author.⁷³ From this one also derives the story of the stored program concept and the need for the stored program, irrespective of the source.

Which was the FIRST business computer? If one uses the video series, “The Machine That Changed the World” as a source,⁷⁴ one would know that the world’s first business computer was the LEO. It was manufactured by the Lyons Catering Company, owner of the corner teahouses in England, who had an inventory problem that was solved by what we now know as “just-in-time” production and delivery. Based on the Cambridge University EDSAC, itself the first productive stored program computer, LEO preceded the UNIVAC I by a year, but without the hoopla and PR provided by Remington Rand. But is it important that this be known by a freshman student? One reason is to ensure that we do not lose touch with the truth that the development of the computer was an international project and not just an American product.

In considering number systems and the development of machines that performed numerical computations, there is an opportunity to discuss, at least briefly, the chronology of the development of the calculator and therefrom the computer. The story of the heritage of the calculator frequently starts with the abacus, which is commonly ascribed to China, or at least the Far East. However, Michael Williams points out that prior to the visit of Marco Polo, there is no record of these machines in the Far East, while there is evidence of similar machines in the Roman era.⁷⁵ Our propensity to ascribe the abacus to Far East countries is perhaps based on the recent memories of their use (and current use in some places) in those countries. The fact that different versions of the abacus can be ascribed to different countries, such as the Swan Pan to Japan and the Soroban to Russia, does not necessitate the belief that these were their countries of origin.

Models of Computation

Abstract machines are easy to ascribe to their originators, and this is one more opportunity to teach something of our history. In our version of this first course each of these studies has an accompanying emulator that the students may use to understand their capabilities and uses. The study of compilers as a tool of computer scientists includes the study of Finite State Machines for lexical analysis, and simple syntactic analyzers that are driven by context-free languages as specified in Backus-Normal Form (BNF), sometimes also known as Backus-Naur Form. The contribution of McCulloch and Pitts in developing the ideas of Finite State machines as a model of nervous behavior in the brain is a prime example of how we have

⁷³ M.D. Godfrey and D.F. Hendry, “The Computer as von Neumann Planned It,” *IEEE Annals of the History of Computing* 15/1 (1993): 11-21.

⁷⁴ I primarily use this series as a substitute when I am away from my institution.

⁷⁵ Michael R. Williams, *A History of Computing Technology*, second edition (New York, 1997).

assembled the foundations of computer science from a variety of sources, and conversely the applicability of computer technology to other fields.⁷⁶ Similarly, the concepts of syntactic analysis combined with the formalization of the descriptive systems for languages is a prime example of the use of linguistic techniques in resolving the problems of understanding language, both natural and synthetic.

The history of programming languages—starting from the early determination to develop “automatic programming” (a goal still not achieved), through the concept of the compiler, and eventually to the demonstration of the practicality of high-level programming languages through FORTRAN and COBOL—is an important lesson for beginning students.⁷⁷ While providing laboratory experiments in “programming” a Finite State Machine, or analyzing strings through a syntactic analyzer driven by a syntactic specification written in BNF, provides the kind of hands-on activity that computer science students prefer, the historical backgrounds further help to solidify the concept in their memory.

Turing Machines

Following up on the claim that Alan Turing (see Figure 3 below) was one of the five original inventors of the computer is a study of the “Universal Machine” that he prescribed in his 1937 paper on computable numbers.⁷⁸ While some emulators of the early machines may possibly exist, the opportunity to program an original computer, limited as it is, is of great interest to our modern students. The opportunity to examine this machine and to show that Turing’s claim to universality is important to the understanding of the foundations of the field, even if this initial exploration is minimal. However, the study of abstract machines is a meaningful event in the Grand Tour with the understanding that a later course will follow up on these basics. Study of the Turing Machine provides the occasion to discuss the life story of Alan Turing and his many other contributions to the field of computing and to other fields such as biology.⁷⁹ Not the least of his work was the code-breaking at Bletchley Park during World War II. A one-hour NOVA video program entitled “The Strange Life and Times of Alan Turing” is available, which can provide details beyond the capabilities of most teachers.

Figure 4 below shows the front face of the Turing Machine emulator used in this course programmed to “sort” a string composed of characters from the set $\{0,1\}$. While this particular emulation is limited in its capabilities, including being limited to ten instructions, it is capable of demonstrating most of the fundamental operations carried by a modern computer such as addition, complementation, and so on.

⁷⁶ Warren S. McCulloch and Walter Pitts, “A Logical Calculus of the Ideas Immanent in Nervous Activity,” *Bull. Math. Biophysics* 5 (1943): 115-33.

⁷⁷ Richard L. Wexelblat (ed.), *The History of Programming Languages*, (New York, 1981); and Thomas J. Bergin and Richard G. Gibson (eds.), *The History of Programming Languages – II* (New York, 1996).

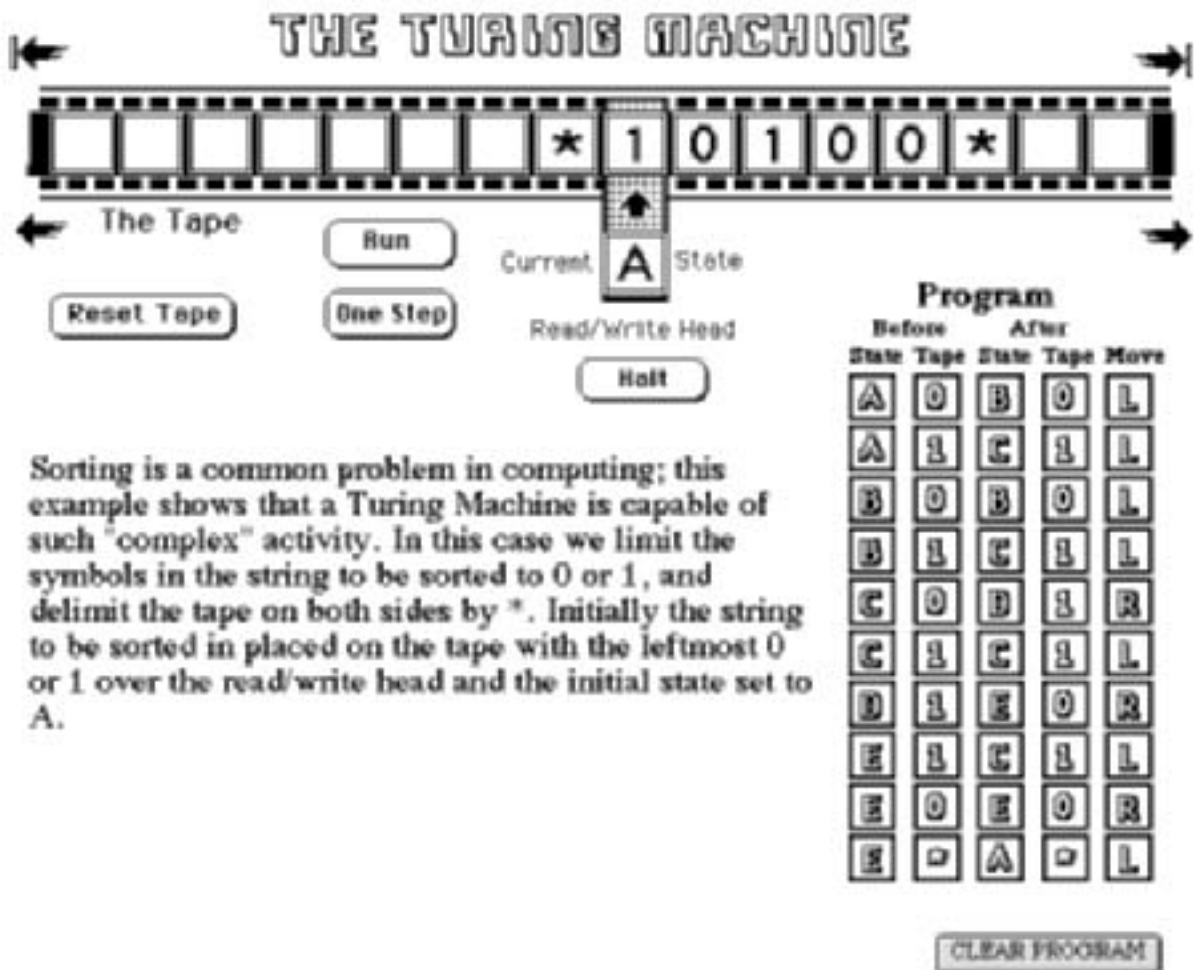
⁷⁸ Alan M. Turing, “On Computable Numbers with an Application to the Entscheidungs-problem” *Proc. London Math. Soc.* 42 (1937): 230-65. This paper defined the concept of the Universal Machine.

⁷⁹ Andrew Hodges, *Alan Turing: The Enigma*, (New York, 1983).

Figure 3. Biographical Vignette for Alan Turing

Turing, Alan Mathison, 1912–54, British mathematician and computer theorist. While studying at Cambridge Univ. he began work in predicate logic that led to a proof (1937) that some mathematical problems are not susceptible to solution by automated computation; in arriving at this, he postulated a universal machine, now called a Turing machine, that was the theoretical prototype of the electronic digital computer. After completing a Ph.D. at Princeton Univ. (1938), he returned home to England, where, during World War II, he was instrumental in deciphering German messages encrypted by the Enigma cipher machine. After the war, he helped design computers, first for the British government (1945–48) and then for the Univ. of Manchester (1948–54). During this period, he produced a body of work that helped form the basis of the newly emerging field of artificial intelligence; among his contributions was the Turing test, a procedure to test whether a computer is capable of human-like thought. He committed suicide shortly after being arrested for a homosexual offense.
 Source: infoplease.com, <http://www.infoplease.com/ce6/people/A0849730.html>

Figure 4. Turing Machine



Beyond the First Course

With a basis of using history in the first course as a “humanizing” element in the teaching of technology, it is appropriate to be able to follow up on this in subsequent courses. After all, the introductory course is simply the “Grand Tour,” and thus students will expect the details to be exposed in the ensuing courses. This presumes that besides the teachers of the introductory course, faculty in other courses are familiar with the history of the field. This raises the question of whether teachers of technology need to be at least conversant with the history of the technology they teach. For several decades computer science has ignored its history in favor of a forward-looking approach that ignores the postulate that those who ignore history are doomed to repeat it. While the professional organizations, primarily the Association for Computing Machinery (ACM) and the IEEE Computer Society, have contributed to the preservation of our history in many ways, the coverage of the field is still “spotty.” ACM organized several conferences in the fifteen years after 1978 entitled “The History of .”, the most famous of which were the History of Programming Languages conferences in 1978 and 1993, but there is still a need to look back at the histories of (say) operating systems, human computer interaction, or of specific subjects such as object-oriented programming. Moreover, most specialized textbooks do not contain a section on the history of the subject; where it is included it often, unfortunately, contains many of the myths discussed above, simply because the authors do not have ready access to original materials or documents.

One solution to this problem is for the computer historians in departments of computer science to organize a “while you are away” program as teacher substitutes, with the understanding that they will give a lesson on the history of computing relative to the course topic. By videotaping the presentation or providing a set of slides/overhead foils, this component can become a permanent part of the course. Working with the instructor can also identify the points on the curve of perceived complexity of development from which teaching points can be extracted to better create a course plan using history to teach the topic.⁸⁰

The Power of Emulators

Though not an organized activity, there is a movement to develop emulators of long past machines that are potential learning devices for students in a variety of courses. Emulators were initially used within the computer industry to permit users who were experiencing a change in equipment, frequently not of their own choosing but sometimes as a result of the obsolescence of their equipment, to continue to use their software on the new machine. In fact there were anecdotes about some software systems relying on a cascading series of emulators so they could still run software for a machine dating back five generations on their current equipment!

Apart from these commercial applications and emulations of simpler abstract machines, the emulation of the EDSAC by Martin-Campbell Kelly for the Macintosh (classic) in 1988 was one of the first to be used in the educational field. The EDSAC (see Figure 5) had a particularly small instruction set using one character op-codes and a set of well preserved programs from 1947, the year in which the EDSAC was installed. This teaching device provides hands-on experience with the first productive stored program computer, but also shows the problems associated with program and data input (and output) that cannot be explored in modern machines that are overloaded with virtual environments to protect the user from the need to

⁸⁰ J.A.N. Lee, “Perceived Complexity: A Conjecture Regarding the Life Cycle of Computer-Related Products,” *IFIP World Computer Congress 2000* (Beijing, China, August 2000).

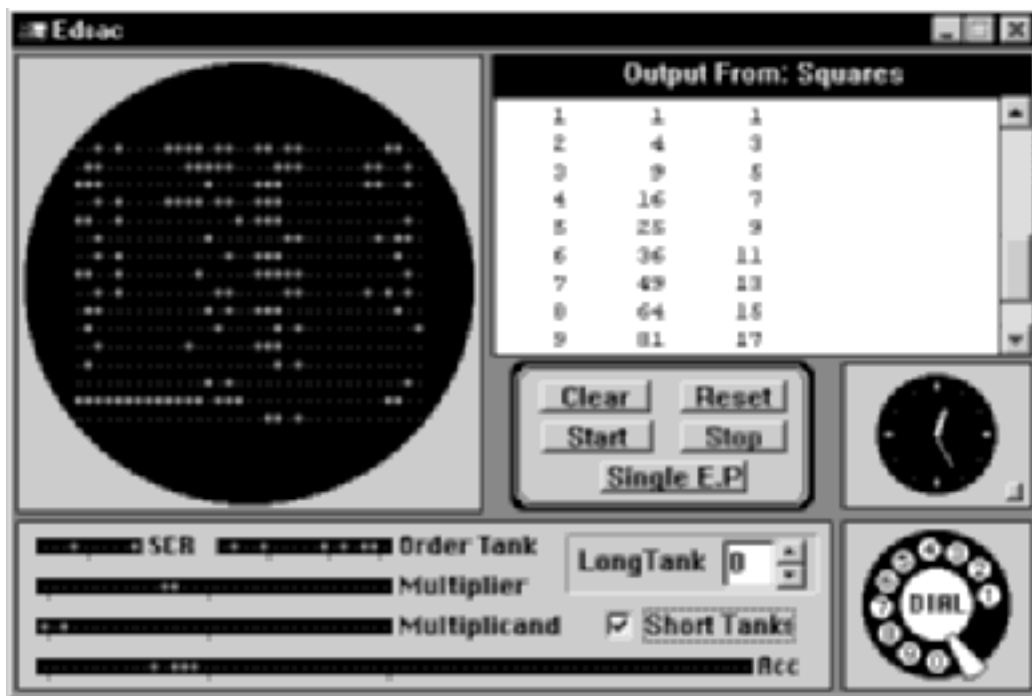
program at the machine level. Since that time many other emulators have been developed, many of them executable on the WWW, as listed in Appendix B.

Ethics in History (or History in Ethics)

History is a source of case studies for use in courses related to ethics and social impact. While the community interested in computer ethics has developed a large number of scenarios for use in their courses, the reality of a truly historical case provides a link to the world that is not found in synthesized examples. Examples include studying the questions:

- Did John Mauchly really base the arithmetic unit of the ENIAC on the ABC?
- What was the impact of the 1956 IBM consent decree on business practices in the computing field?
- Could the Therac 25 problems have been avoided?
- Should computer pioneers have been more protective of their inventions?

Figure 5. The EDSAC Simulator Developed by Martin Campbell-Kay



Department of Computer Science, University of Warwick, England. Copyright, 1996-2001.
Source: <http://www.dcs.warwick.ac.uk/~edsac/>

The collection of cases is another of those underdeveloped aspects of the history of computing. This is primarily the result of our being overly protective of the sensitivity of developers whose projects were not successful. Our archival journals rarely contain technical articles on why an article failed or did not succeed, and locating the original documentation, if any, is nigh on impossible. The archives of the Charles Babbage Institute at the University of Minnesota are one potential source of cases, but so far their funding for detailed examination of the deposited materials has been minimal. Surely the material is there! But we cannot expect teachers to go looking for it; that is the task of historians.

Conclusion

Using history to improve undergraduate education in computer science is a given among computer historians, but this will be a difficult task until:

1. Historians with an education bent extract appropriate resources from archives and museums;
2. The history of components of the computing field is extended to cover the common subjects in the curriculum—particularly in the areas within “Curriculum 2001” that do not currently include a segment on their history;⁸¹
3. Readily accessible and well-assimilated resources are available that will permit potential user-teachers to easily incorporate history into their syllabi; and
4. History as a learning support discipline is accepted by the teaching community.

Overall, we must convince funding agencies such as NSF-DUE and other foundations that support the development of educational resources to fund projects that will identify resources and develop lesson plans in support of the computer science curriculum.

John A.N. (JAN) Lee is professor emeritus of computer science at Virginia Tech having spent 47 years in teaching and research, starting in civil engineering in the 1950s and converting full time to computer science in 1964. As a civil engineer he was involved in the design of several structures around the world that are well known, including the Sydney Opera House and the Firth of Forth Road Bridge. Before getting deeply involved in the history of computing he was involved in national and international standards, leading the work on the standards for PL/I, BASIC and Ada. In the last twenty years of teaching he was a champion for the introduction of history, ethics, and social awareness into the computer science curriculum. He is a fellow of the Association for Computing Machinery and of the Institute for Distributed and Distance Learning. He is the author of eight books and over 400 articles. *Author’s address:* 4247 Horseshoe Loop, Dublin VA 24084-5863 USA. Email: janlee17@verizon.net.

⁸¹ ACM/IEEE Computer Society, “Computing Curriculum 2001” (Steelman Report), <http://www.computer.org/education/cc2001/steelman/cc2001/index.htm>.

Appendix A. Concepts Covered in the Introductory Curriculum

(From ACM/IEEE Computer Society, "Computing Curriculum 2001" (Steelman Report), <http://www.computer.org/education/cc2001/steelman/cc2001/index.htm>, Chapter 7)

<i>Concept</i>	<i>Description</i>	<i>Associated activities</i>
Algorithmic computation	Algorithms as models of computational processes; examples of important algorithms	Read and explain algorithms; reason about algorithmic correctness; use, apply, and adapt standard algorithms; write algorithms
Algorithmic efficiency and resource usage	Simple analysis of algorithmic complexity; evaluation of tradeoff considerations; techniques for estimation and measurement	Estimate time and space usage; conduct laboratory experiments to evaluate algorithmic efficiency

Programming fundamentals

<i>Concept</i>	<i>Description</i>	<i>Associated activities</i>
Data models	Standard structures for representing data; abstract (described by a model) and concrete (described by an implementation) descriptions	Read and explain values of program objects; create, implement, use, and modify programs that manipulate standard data structures
Control structures	Effects of applying operations to program objects; what an operation does (described by a model); how an operation does it (described by an implementation)	Read and explain the effects of operations; implement and describe operations; construct programs to implement a range of standard algorithms
Order of execution	Standard control structures: sequence, selection, iteration; function calls and parameter passing	Make appropriate use of control structures in the design of algorithms and then implement those structures in executable programs
Encapsulation	Indivisible bundling of related entities; client view based on abstraction and information-hiding; implementer view based on internal detail	Use existing encapsulated components in programs; design, implement, and document encapsulated components
Relationships among encapsulated components	The role of interfaces in mediating information exchange; responsibilities of encapsulated components to their clients; the value of inheritance	Explain and make use of inheritance and interface relationships; incorporate inheritance and interfaces into the design and implementation of programs
Testing and debugging	The importance of testing; debugging strategies	Design effective tests; identify and correct coding and logic errors

Computing environments

<i>Concept</i>	<i>Description</i>	<i>Associated activities</i>
Layers of abstraction	Computer systems as a hierarchy of virtual machines	Describe the roles of the various layers in the virtual machine hierarchy
Programming languages and paradigms	Role of programming languages; the translation process; the existence of multiple programming paradigms	Outline the program translation process; identify at least two programming paradigms and describe their differences
Basic hardware and data representation	Rudiments of machine organization; machine-level representation of data	Explain basic machine structure; show how different kinds of information can be represented using bits
Tools	Compilers, editors, debuggers, and other components of programming environments	Use tools successfully to develop software

Appendix B. Emulators of Historical Systems

(from: <http://ei.cs.vt.edu/~history/emulators.html>)

Abstract Machine Emulators:

Napier's Bones and Chessboard calculators

Finite State Machines

Student version by JAN Lee

An Interactive System for the Exploration of Finite State Machines by Matt Chapman (UK).

Turing Machines

Student version by JAN Lee

A Turing Machine Emulator by Paul A. Queior, Brandeis University

Several emulators of the Manchester Mark I:

<ftp://samson.kean.edu/pub/leew/madm.zip> -- an early, DOS-based version, in C

<ftp://samson.kean.edu/pub/leew/wmadm.zip>

<ftp://samson.kean.edu/pub/leew/wmadmsrc.zip> -- a later, Windows-based version, in Delphi

<ftp://samson.kean.edu/pub/leew/ssem/> -- the most recent version, written as a literate program in Java (1.1) using noweb and pretzel, written in honor of the 50th Anniversary

Other emulators:

An excellent on-line Applet simulator of a three-wheel enigma by Andy Carlson, ATT Labs, UK.

Andy Carlson also has a C++ model of the Bombe machine used at Bletchley Park.

The Enigma Machine by Russell Schwager, Johns Hopkins University Computer Science Department.

A very nice EDSAC emulator for PCs and Macintoshes by Martin Campbell-Kelly, University of Warwick.

Though not truly a machine, this CP/M Emulator is an application to emulate CP/M-80 ver2.2 for the Macintosh from Japan.

Emulation.net is a collection of emulators for the Macintosh includes not only many emulators for game playing machines, but also the following systems:

Amstrad CPC

Amstrad PCW

Apple I

Apple II

Apple III

Atari 800

Atari ST

BBC Micro

Commodore 64

Commodore Amiga

CP/M

Edsac

IBM PC

IBM Series 1

Macintosh
Memotech MTX
MIPS R2000
MO5
MSX
Oric
PDP-8/E
SAM Coupé
Sinclair QL
Sinclair ZX81
Sinclair ZX-Spectrum
Thomson TO8
TI99
TRS-80
TRS-80 Color Computer
VAX
VIC-20

A Simple Computer Emulator by Beach Internet Studios.
How to Write a Computer Emulator by Marat Fayzullin.

10. History of Computing for Computer Science and Information Systems Students⁸²

Thomas J. Bergin
American University in Washington

Jack Hyman
SRA International

Abstract

This paper discusses a one-semester course in the History of Computing for computer science and information systems students, taught by the author at the American University in Washington, D.C. A syllabus is included, along with a discussion of how such a course evolved, course requirements, and the use of a “show-and-tell” teaching strategy that makes use of computer artifacts. Readers are also advised to read a companion paper, “Key Resources in the History of Computing” (chapter 26, this volume) and to visit the Computing History Museum website (URL: www.computinghistorymuseum.org), which contains PowerPoint® slides for the lectures in this course and identifies Web and other resources that can support the teaching of courses on the history of computing.

In fall 1988, I created an elective course for computer science and information systems majors at American University.⁸³ In my undergraduate and graduate studies, I had taken courses that exposed me to the history of my major subjects. For example, as an undergraduate English major (at the University of Maryland), a course in the History of the English Language was a requirement. Although my Masters work did not require a “history” course, the doctoral program in Public Administration (at American University) had required courses on the evolution of administrative and management theory. In each case, the history course gave me insights into why things were as they were, and a broader context in which to view new subjects. Although I was never proficient in Old English or Middle English, reading *Beowulf* and Chaucer’s *The Canterbury Tales* in the original were special experiences—which were accompanied by the instructor’s comments and interpretations. Thus, it seemed reasonable to expect that undergraduate and graduate students in computer science and information systems should learn something about the origins of computing!

CSIS-550: The History of Computing⁸⁴

CSIS-550, The History of Computing, was a one semester, upper-division course that covered the history of computing from the abacus to the present.⁸⁵ To prepare for the course, I

⁸² This paper was written in the first person by the first author; the second author worked on the website supporting this course and has served as its webmaster since 2001; he also did the graphics for this paper.

⁸³ At this time, we had about 180 undergraduates (about 120 Information Systems majors and 60 Computer Science majors) and about 200 graduate students. The majority of the students taking the History of Computing class were Information Systems majors, but the course was open to all students on campus and attracted a fair number of business students over the years.

⁸⁴ At American University, the Information Systems curriculum began in the school of Public Administration in the early 1960s; the Computer Science curriculum was part of the Mathematics department until 1987, when the two curricula were merged into the Computer Science and Information Systems (CSIS) department of the College of Arts and Sciences (CAS). A separate IS curriculum was supported in the Kogod College of Business Administration (KCBA). In 2002, the IS curriculum was transferred to KCBA and the Computer Science curriculum moved to the Physics and Audio Technology Department.

contacted Mike Williams, Brad Burke, and Paul Ceruzzi and sought their advice. Mike Williams sent me a syllabus that he used at the University of Calgary.⁸⁶ Since Brad Burke taught nearby at the University of Maryland Baltimore County, I arranged to visit him. Brad generously gave me a copy of his syllabus and some insights into how to go about building a history of computing course.⁸⁷ Lastly, Paul Ceruzzi, who also lives in the Washington area, gave generously of his time and knowledge to help me organize the course.⁸⁸ Indeed, Mike and Paul visited the class over the years as guest lecturers, as did a few “pioneers” such as Carl Hammer, Jean Sammet, Keith Reid-Green, and George R. Trimble, Jr.

Due to time requirements, the level of detail is non-uniform, meaning that some topics are covered in more detail in this course than other teachers might like. I believe that some subjects, such as Charles Babbage, the ENIAC, and the Altair, must be included in a course on the history of computing. The inclusion of other topics such as the abacus, punched card accounting, and the IBM System 360 depends on the institution and course objectives, as well as the instructor’s experience and comfort level with the material.

Since I started teaching this course in 1988, the course has been based on *A History of Computing* by Michael R. Williams.⁸⁹ In the mid-1990s when this book was out of print, I used two books: *Reckoners, The Prehistory of the Digital Computer, from Relays to the Stored Program Concept, 1935-1945*, and *Computing Before Computers*.⁹⁰ With a small university grant, I was able to purchase twenty copies of each of these books for class use. When the IEEE edition of the Williams book was available, I adopted it as the course textbook.

As can be seen from the syllabus, there are also a number of journal and magazine articles that can be used as supplementary material. This supplementary material was put on reserve in the Bender Library, American University. Currently, there are many high-quality books on the history of computing that could be used as a course textbook. These are referenced on the Computing History Museum website at the following URL: <http://computinghistorymuseum.american.edu/teaching/Textbooks.htm>. Additional information is contained in the companion paper in this volume, “Resources for a Course on the History of Computing” (see chapter 27).

The Syllabus

The syllabus contained in Appendix A tracks the Williams text, with additions to cover topics such as punched card technology, minicomputers, and microcomputers. It is broken into sections: Description and Purpose, Course Objectives, Course Requirements, Grading, Required Reading, Recommended Reading, Writing Quality, Plagiarism, and the Class Schedule, which lists the lectures by dates along with any additional recommended reading.

⁸⁵ At American University, 500-level courses are restricted to juniors, seniors, graduate students, and others at the discretion of the instructor—who allowed any interested student to take it.

⁸⁶ After retiring from the University of Calgary, Dr. Michael Williams accepted the position of Chief Curator at the Computer History Museum in Palo Alto, CA.

⁸⁷ Dr. Colin (Brad) Burke is Emeritus Professor of History, University of Maryland, Baltimore County.

⁸⁸ Dr. Paul Ceruzzi is Curator of Space History at the Smithsonian’s Air and Space Museum and a contributor to this volume.

⁸⁹ See Michael R. Williams, *A History of Computing Technology* (Englewood Cliffs, NJ, 1985). In 1997, the IEEE Computer Society published a second edition, which is the one used in the syllabus.

⁹⁰ Paul Ceruzzi, *Reckoners, The Prehistory of the Digital Computer, from Relays to the Stored Program Concept, 1935-1945* (Westport, CT, 1983) and W. Aspray et al, *Computing Before Computers* (Ames, IA, 1990).

The course contains fourteen lectures as follows, all of which may be found at the Computing History Museum website:
(<http://www.computinghistorymuseum.org/teaching/lectures/lectures.html>).

1. Introduction and Numeration
2. Preconditions to Computing: Early Aids to Calculation
3. *Charles Babbage* (and Joseph Marie Jacquard)
4. Herman Hollerith & Early Accounting Machines*
5. Information Appliances and Office Appliances*
6. Analog Devices and Mechanical Monsters
7. Electronic Revolution, John Atanasoff, and ENIAC
8. Early Stored Program Computers⁹¹
9. Mainframe Computing
10. History of Transistor and Digital Equipment Corporation^{92*}
11. The Computer Industry, Year by Year^{93*}
12. Micro-computing^{94*}
13. Personal Computing*
14. Presentation of Graduate Research Papers
(* These topics are outside the scope of the Williams textbook.)

As can be seen from the above list, the course evolved over time. If we look at the changes to the discipline of computing since the first edition of the Williams textbook (1985) to the present, we can see the need for additional topics and reading materials.⁹⁵

When I first taught the course, I stayed fairly close to the contents of the text. As I read more material and gained confidence, I took the course in new directions. For example, my inquiries to (then) Digital Equipment Corporation resulted in my acquiring multiple copies of *Digital at Work: Snapshots from the first thirty-five years*, edited by Jamie Parker Pearson (Bedford, MA, 1992). Jamie also kindly gave me copies of the slides used in the presentation that resulted in this book.⁹⁶ With these resources, plus some hardware modules I had collected over the years, the lecture on mini-computing was on solid ground. Figure 1 shows a DEC module, which can be found at many computer swap meets and similar venues.

⁹¹ The website lists the title of this lecture incorrectly as *Early Stored Computer Programs*, quite a different subject!

⁹² After all these years, I have just noticed that this lecture is incorrectly listed as "Digital Electric Computer Company." Along with the previous note, this indicates a lack of careful proofreading on my part, for which I apologize.

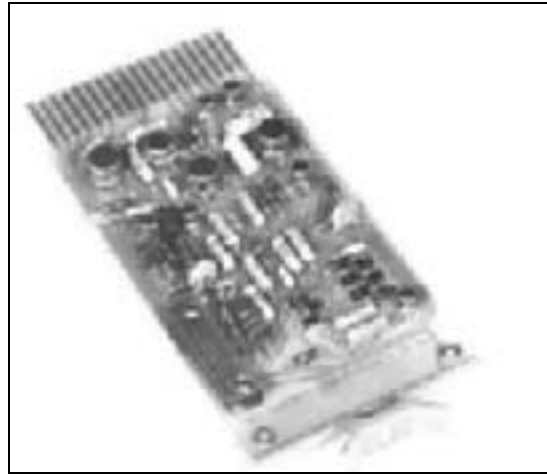
⁹³ Although there is a lecture on the website with this name, this night is used for student presentations; the lecture was prepared for other purposes and is included on the site in case someone can use it.

⁹⁴ For almost all of the history of the course, Jean Sammet was a guest lecturer on "The Evolution of Programming Languages," a topic in which she has no peer.

⁹⁵ *A History of Computing Technology* ends with Chapter 9, "Later Developments," which discusses the early machines of IBM and early supercomputers (i.e., in the early 1960s).

⁹⁶ Some of these slides were used to create the *PowerPoint*® slides for the "Transistor and Minicomputers" lectures, including Figure 1.

Figure 1. Digital Equipment Corporation (DEC) Module



Toward the end of the 1990s when personal computers, as tools, were being used in a variety of university courses, I decided to add one (and then two) modules on the origins of micro-computing and personal computing. By then, the literature on micro-computing and personal computing was both rich and robust—a necessary ingredient for instructors! The web was also a rich source of information and graphics for creating an interesting set of *PowerPoint*® slides to illustrate the lectures. Figures 2 and 3 show two images from these lectures.

Figure 2. Front Cover of *Popular Electronics*, January 1975

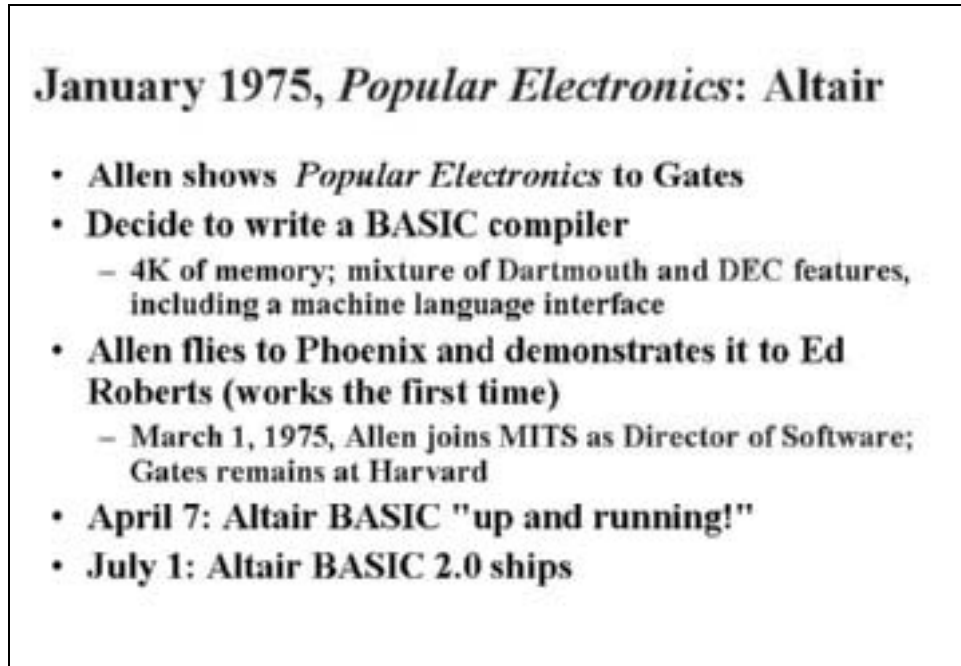


Source: <http://www.blinkenlights.com/pc.shtml>

I strongly believe that such pictures have enormous power in terms of relating to students what was going on. Students can see how simple these publications were and how they lacked the fancy covers, color graphics, and overall sophistication of more recent materials.

Indeed, to provide these images, and support student learning, the *PowerPoint*® slides were reproduced in handout format (six to a page) in black and white (see Appendix B).

Figure 3. Sample Slide on Altair History



January 1975, *Popular Electronics*: Altair

- **Allen shows *Popular Electronics* to Gates**
- **Decide to write a BASIC compiler**
 - 4K of memory; mixture of Dartmouth and DEC features, including a machine language interface
- **Allen flies to Phoenix and demonstrates it to Ed Roberts (works the first time)**
 - March 1, 1975, Allen joins MITS as Director of Software; Gates remains at Harvard
- **April 7: Altair BASIC "up and running!"**
- **July 1: Altair BASIC 2.0 ships**

Comments

Although most of the syllabus is self explanatory, I have included some thoughts to guide instructors contemplating such a course. I invite them to examine the website thoroughly (<http://computinghistorymuseum.american.edu/>) and to communicate with me via email (tbergin@american.edu). Please note: the *PowerPoint*® slides for the lectures are there to be used—interested faculty can view them to see if they meet their needs, and then download and modify them for content and personal style. In the past two years, a number of instructors from other universities have requested permission to use the slides in their classes. Although permission is not required, we do ask that they: 1) tell us that they are using the materials, 2) use “normal academic courtesies” with respect to citing the Computer History Museum as the source of the materials, and 3) provide us with feedback so that we might correct mistakes, add additional materials, and modify existing ones. Figure 4 shows the top of the lectures page, showing the *View* and *Download* options.

Course Objectives

Looking back on more than fifteen years of teaching this course, I realize that the environment and objectives of each institution, academic department, and faculty are different. Thus, what is appropriate for a research-oriented graduate program in computer science will be significantly different from the content of an undergraduate course in information systems (typically located in a business school). However, the major goal of my course was to excite students about the history of computing without regard to major or academic level.

Method of Instruction

Although many readers will think this is a non-topic, I believe there are a number of things that instructors can do to enrich the student's experience and make the course exciting. The Computing History Museum website identifies and contains pictures of a large number of artifacts. I firmly believe that the use of artifacts in the classroom, "*show-and-tell*" if you will, is a powerful teaching tool that makes this sometimes invisible and obscure technology real!

Figure 4. Lectures in the History of Computing



The screenshot shows the website for the American University Computing History Museum. The main heading is "Lectures in the History of Computing". A navigation menu on the left lists: Introduction, Syllabus, Lectures, Class Materials, Student Projects, Discussion, Bibliography, and Resources. The main content area contains the following text: "This page contains Professor Tim Bergin's lectures from the class CSIS 550, History of Computing. We are providing two formats of lecture slides - MS PowerPoint for download and HTML to view. You can always come back from the slides to this page by clicking on the Back button on the browser window." Below this text are two lecture entries. The first entry is titled "Overview of the History of Computing" and includes a thumbnail image of a person at a computer. The second entry is titled "Lecture 1" and includes a thumbnail image of a slide with the word "NUMERATION" and a diagram of a binary number. Both entries have a bullet point with a link to "View | Download".

Source: <http://www.computinghistorymuseum.org/teaching/lectures/lectures.html>

Indeed, I believe that only by referring to previous technology can instructors make the present have meaning. A recent advertisement for a Dell personal computer stated that it had an "Intel®Pentium®processor with HT technology, (operating at) 3.40 GHz and (using an) 800 MHz front side bus and an Intel®875P chipset with ICH5R." How does one explain what all that means—to the satisfaction of any student? If students can learn how older, simpler, technologies like a "core" memory works, then the newer technologies and capacities become understandable.

Requirements

The course has a midterm examination and two projects: a biography paper and a period paper. Graduate students have a research paper. The midterm examination is a

take-home exam that basically requires students to review the textbook, lectures and outside reading and compose lengthy replies to a long list of questions. Most students complained that the exam took them five or six hours to complete, but admitted that they learned a lot in pulling their answers together. The take-home nature of the exam allows me to demand more of the students and to give them an additional lecture (by not wasting a class with an in-class exam). A sample midterm examination is included as Appendix C.

The *biography paper* examines the life and contributions of a significant person in the evolution of computing. The website contains some suggested books of biographies that students can use to get their research started (i.e., they contain short biographies and references to sources). Additional sources are also posted on the website.

Since it is impossible to cover the period from the early 1950s to the present with any depth, students are expected to examine *Datamation* and *Computerworld* for a specific year (as assigned) and write a *period paper* describing the most important computer news for that year.⁹⁷ Students are encouraged to scan the tables of contents of the issues for that year and look for important hardware, software, or industry topics. Students are also advised to look at the advertising. The period paper is presented to the class, and students were required to introduce their year by identifying the major topics in the periodical literature (i.e., from *Time*, *Newsweek*, *Business Week*, etc.)⁹⁸

Graduate students are also expected to write a *research paper* that takes an element of present computing and traces it to its origins. We all take (external storage) *media* like CD-ROMs for granted, but what were the first external storage systems and how did they work? What was the capacity of the first tape or disk? And who were the people who built the first storage systems? Graduate students learn a great deal by researching their subject technology; the undergraduates learn from the presentation of these research papers on the last night(s) of class. Some examples of research papers are at the following URL:
http://computinghistorymuseum.american.edu/teaching/projects_research.htm.

Recommended Reading

Although the textbook is the main source of information, there is an increasingly rich source of materials available to instructors and students. Among the most important is the *IEEE Annals of the History of Computing* (<http://www.computer.org/annals/>). As stated in the syllabus, this is the principal scholarly publication in computing history, and all CS and IS students ought to have at least heard of it! As part of the midterm examination, every student has to read and comment on at least one article in *Annals*.

⁹⁷ Although there were other general-interest periodicals during these years, restricting student research to *Datamation* and *Computerworld* provides some continuity between student presentations and makes it easier for the instructor to master the material. *Datamation* has published issues to commemorate milestones in their publishing history that contain historical timelines of events: 25th Anniversary in the September 1982 issue, "Reaching 30 Years" in the 15 September 1987 issue, and "40 Years of Computing" in the 15 March 1991 issue.

⁹⁸ Student presentations use *PowerPoint*® or other presentation software and are planned for eight to ten minutes, including questions.

For Instructors Only

Another source of learning is the reading of out-of-date periodicals like *Datamation*, *Computerworld*, and other professional and trade publications. Reading about “current events” thirty years later or about the early attempts to build external storage devices should result in some real insights for interested and motivated students (and faculty, too!).

Since computing is more than fifty years old, the major professional and technical publications have produced special issues that reflect on parts of this history. For example, *Communications of the ACM* celebrated its 40th anniversary in October 1987 (Volume 30, Number 10) with a special issue devoted to the history of computing. ACM’s 50th anniversary issue (February 1997, Volume 40, Number 2) examined “The Next Fifty Years” and contained essays by forty-three luminaries in the field. *IEEE Computer* celebrated its 40th anniversary in September 1991 by looking at “The Promise of the Next Decade.” The 50th anniversary issue of *IEEE Computer* (October 1996) contained “Essays on Fifty Years of Computing” by leaders in the field, as well as a detailed timeline that should be very useful to instructors new to the material.

The ACM timeline can be found at: <http://www.acm.org/top/tl/index.html> and the IEEE Computer Society’s timeline can be found at: <http://www.computer.org/history/>. The IEEE also has a new Virtual Museum at: <http://www.ieee-virtual-museum.org/>. All of these sites deserve extended visits and are very helpful in putting pieces of history in context.

The Lectures

As stated above, the course has fourteen lectures. *PowerPoint*® slides for each lecture are available at: <http://computinghistorymuseum.american.edu/teaching/lectures/lectures.htm>. Each lecture also has a list of “show-and-tell” items as the last page (which is not reproduced for the student handout). If you are serious about having a course in the history of computing (and you intend to offer it on a continuing basis) then it is worth the time and effort to look for, and acquire, a collection of artifacts that can be used in the classroom. For example, for the first lecture on “Numeration” I use the following artifacts: sample hieroglyphic tablets, a prayer rug, a recreation of an astrolabe, and an egg timer. I found the hieroglyphic tablets at an art fair in Palo Alto. The prayer rug was a gift from a student from Kuwait, who took the course the first time it was offered and thought the prayer rug would help explain the need for mathematics in Muslim countries and the importance of the contributions of Al-Khowarizmi.⁹⁹ I purchased the astrolabe from the Franklin Mint in the mid 1980s, after seeing an advertisement in *Scientific American*. An egg timer should not be hard to find, even in a “microwave world.”

Instructors can purchase or make replicas of materials for classroom use. Although a full description of my artifact collection (acquired or fabricated) is beyond the scope of this paper, I will identify a few significant ones. Please refer to the companion paper, “Resources for a Course on the History of Computing” (see chapter 27, this volume). The Virtual Museum Tour (<http://computinghistorymuseum.american.edu/museum/index2.htm>) at the Computing History Museum website contains pictures of some of these artifacts.

⁹⁹ The name of this 8th century Muslim scholar is spelled many ways. I have adopted the full name and spelling used by a graduate student who wrote a research paper on *Abu Jafar Mohammed Ibn Musa Al-Khowarizmi*, from whose name we get our word “algorithm” and from whose writings we get our word “algebra.”

The *Abacus* can be found in at least three versions: the Chinese *Swan Pan*, the Japanese *Soroban*, and the Russian *String Abacus*. Over the years, I have acquired more than ten Swan Pans and Sorobans from numerous sources—mostly through donations from friends and colleagues. Replicas of many devices can also be made with a little effort and ingenuity.¹⁰⁰ Although few people have replicas of Schickard or Pascal-like calculators, everyone can find books of tables, meter sticks, and slide rules!¹⁰¹ Just ask someone with gray hair if they have any of these things! As part of the lecture on “Early Aids to Calculation” there is a “laboratory” in which students do a number of exercises using an abacus. I also “create” a slide rule using two meter sticks (easier than yardsticks!) and a log of the numbers 1 through 10.

Among the more interesting artifacts from a student perspective is the use of the Gelosia method for multiplication and the invention of Napier’s bones by John Napier (1550-1617). The replica in the Museum was made in the University’s wood shop, but I had made one many years before from balsa wood for practically no cost. The Gelosia method can be demonstrated using handouts contained in the lecture slides.

For the lecture about Jacquard, I made a small frame (about 6” x 18”) with threads going in both directions. After an explanation of Jacquard’s machine (with photographs) students can see how the machine “hooks and lifts” selected threads to create a pattern. For the Babbage lecture, a few visits to an antique shop will probably yield a skeleton key, some printers’ stereotypes, and books of logarithms, all long ignored by most people.

The Smithsonian’s *Information Age* exhibit and Steven Lubar’s book, *Information Appliances* gave me the idea to extend the boundaries of the course to include a short history of communications and to discuss early office equipment—predecessor technologies to current word processing, accounting, and email capabilities.¹⁰² After a few visits to antique shops in Sussex County, Delaware, I found a telegraph key; and a telegraph receiver was purchased for a few dollars in an antique store in Maine. An automobile battery charger and some two-strand wire make it all work.¹⁰³ One of the most interesting learning experiences for me occurred the first time I had a “laboratory” of office appliances. I put ten different types of mechanical calculating machines and two typewriters around the room; each had a short problem for students to do. Students then went around the room trying (playing with?) the various devices. I answered questions and demonstrated use as necessary. It was interesting how quickly students realized that a Brunsviga and a Comptometer were used for different types of calculations. Class was supposed to end at 10:30 p.m., but a small cadre of students kept me there till after midnight!¹⁰⁴ This was technology they could see and feel; they could relate this to the computers they used every day. It was like visiting Grandpa’s office!

¹⁰⁰ Recently I have had the opportunity to teach short “mathematics” classes to elementary students. I created a table abacus from a piece of foam-core (30” x 40”) and used wooden disks as markers. The markers are held on the abacus (which is horizontal so the students can see it) with Velcro®. Students follow along using poker (or game) chips on a paper copy.

¹⁰¹ I purchased the Schickard calculator for \$400 US from a retired machinist in the Netherlands; the Pascal-like calculator was offered by the Computer Museum in the early 1980s for a few dollars.

¹⁰² Steven Lubar, *Infoculture, The Smithsonian Book of Information Age Inventions* (Boston, 1993).

¹⁰³ Before this demonstration, I usually ask, “Has anyone ever driven down Telegraph Road?” (in Alexandria, VA). Most students from the Washington area answer “Yes.” The follow-on questions: “Why do you think it is called Telegraph Road?” or “What is a telegraph?” usually brings blank stares or humorous guesses. I make it a point never to ask about Gallows Road (which is also nearby)!

¹⁰⁴ American University had two night-class slots for upper division courses: 5:30 p.m. to 8:00 p.m. and 8:10 p.m. to 10:40 p.m.; the use of artifacts was a definite help in keeping students alert and interested at such a late hour.

For the computer classes, I drew upon my collection of artifacts, but suffice it to say that the older the better. Students liked seeing racks of vacuum tubes from first-generation mainframe computers and plug modules from minicomputers like the PDP-8. Indeed, when I dragged out old personal computers, like the Commodore 64, some student always said, "That was my first computer!" Students also liked seeing (and touching) 10" floppy disks (from a TRS-80 Model II, 1979), 5 ¼" floppy disks (from an Apple II, 1977), and hearing that the Texas Instruments 99-4A, which I bought at Toys R Us for \$299 (less a \$100 rebate!) used a *cassette tape* for storage (introduced in 1980; the rebate was offered on August 6, 1983; the final price in late 1983 was \$49.95).¹⁰⁵ I have come to understand that "modern" people have little or no understanding of history, even the history of *twenty years ago*. Indeed, it is sometimes difficult to remember that most of our freshman students weren't even born twenty years ago!

Other artifacts appropriate for classroom use are for sale on the Web and at computer swap meets. Although it takes time to look for such things, they are invaluable in the classroom.

Conclusion

I hope that this paper doesn't make it sound like collecting all this "stuff" was easy or that it didn't take time. I started collecting things in 1967 in my first computer-related job (as a digital computer systems analyst with the US Veterans Administration). When people would announce their retirement, I would visit them and beg for materials. Later I started visiting antique shops and actively looking for artifacts. It was great fun. The short version story of my collecting affliction can be found in an article published in *Yesterday's Office*.¹⁰⁶ A longer version is posted on the Computing History Museum website under "Museum History."¹⁰⁷

I used the history of computing and computing artifacts in almost every course, and I believe that our history has enriched my students' experiences. Moreover, I believe knowledge of computing history is critical to understanding present technology and the proper use of that technology by computer scientists, engineers, and information system professionals.

In this paper, I have tried to outline my motivations for creating a course in the history of computing for computer science and information systems majors, as well as some of the tools that I used in the classroom. I strongly encourage others to do the same, and to make use of the Computing History website, especially the Teaching pages, to make their efforts easier.

Thomas J. (Tim) Bergin is Emeritus Professor of Computer Science at the American University in Washington, D.C. Dr. Bergin entered the computing field in 1966, as a "digital computer systems analyst (trainee)" with the U.S. Veterans Administration. In 1980, he received an Inter-governmental Personnel Act Fellowship as a Visiting Professor in American University's Center for Technology and Administration. In 1982, he joined AU as an Assistant Professor and Director of the Quantitative Teaching and Research Laboratory, and managed the introduction of personal computers to support faculty and student research. He has authored or edited four books: *History of Programming Languages* (ACM Press, 1996) and *Fifty Years of Army Computing: From ENIAC to MSRC* (Army Research Laboratory Special Report-93, September

¹⁰⁵ Haddock, pp. 119, 103, and 133, respectively.

¹⁰⁶ <http://www.yesterdaysoffice.com/index.cfm?fuseaction=ShowArticle&articleid=36>

¹⁰⁷ <http://www.computinghistorymuseum.org/museum/comphist.pdf>.

2000), *A Microcomputer-Based Primer on Structural Behavior* (with James Letter, Prentice-Hall, 1986), and *Computer-Aided Software Engineering* (Idea Group Publishing, 1993). In addition to his teaching duties, Dr. Bergin created the Computing History Museum at American University, which was renovated out of existence upon his retirement in 2002. To see some of the artifacts mentioned in this article you can visit the Virtual Computing History Museum (<http://computinghistorymuseum.american.edu/>). A story about the creation of the Museum was also featured in an article in *Yesterday's Office*.¹⁰⁸ Dr. Bergin also served as the Assistant Editor-in-Chief of the *IEEE Annals of the History of Computing* from 1995 to 1999 and as Editor-in-Chief from 1999 to 2003. Dr. Bergin also served as historian to the ACM for their 50th anniversary celebration. *Author's address*: 217 Pewter Lane, Silver Spring, MD 20905 USA. Email: tbergin@american.edu.

Jack A. Hyman is a software engineer with SRA International Inc. He has been a member of the Project Management Institute, International Webmasters Association, IEEE, ACM, and The e-Learning Guild. He holds an M.A. in Education & Human Development, Educational Technology Leadership from The George Washington University and a B.S. in Computer Information Systems from The American University. His areas of interests include: usability engineering, content and knowledge management, computing history, and information design. Jack joined the Sloan Research Team as a freshman (in 1998) and played a critical role in the evolution of the site since that time, serving as webmaster since 2001. As a senior research project, Jack created a PowerPoint presentation based on Mitch Waldrop's *The Dream Machine*. *Author's address*: 1400 South Joyce Street, Apt 1609, Arlington, VA 22202 USA. Email: jahyman@comcast.net.

¹⁰⁸ <http://www.yesterdaysoffice.com/index.cfm?fuseaction=ShowArticle&articleid=36>

Appendix A. Syllabus for CSIS-550: History of Computing

CSIS-550 History of Computing

Thursday, 5:30 to 8:00

Thomas J. Bergin, PhD
tbergin@american.edu

DESCRIPTION AND PURPOSE

This course examines the evolution of computers and information systems, from the abacus to the microcomputer. Students will examine the evolution of computing machinery, software and programming languages, and be introduced to important men and women in the history of computing. Class discussions will put the development of computing into an historical context

COURSE OBJECTIVES

1. To gain an understanding of how computing hardware and software evolved, and the "great ideas" in this evolution.
2. To study some of the major technological milestones and examine the lives of some of the more significant men and women in the history of information and computing.
3. To put the evolution of computers in an historical context.
4. To gain some experience doing historical research.
5. To provide a basis for assessing the future of computing and information systems based on a fuller understanding of the past.

METHOD OF INSTRUCTION

As appropriate for a CSIS elective, most classes will be a mixture of lecture, videos, class discussions, and hands on activities. Class members should read assignments before each lecture; the reading of additional material is strongly encouraged. Outside speakers will be invited to discuss their experiences.

Students are strongly encouraged to visit the Smithsonian's **Information Age** exhibit.

Additional material may be found at: <http://computinghistorymuseum.org/>

COURSE REQUIREMENTS

Note: projects must be submitted in paper and machine form so that they can be put on the course/Sloan Research web site.

Biography Paper: will examine the life and contributions of a significant person in the evolution of computing, such as Charles Babbage, Ada Augusta Byron, Herman Hollerith, Alan Turing, etc. Papers may focus on people active in the 1950s and 1960s when you can find them! Papers should focus on an individual's contributions, and/or the process of invention/discovery/service. **Note:** an excellent starting point is Robert Slater, *Portraits in Silicon*, MIT Press, 1987. See also: J.A.N. Lee, *Computer Pioneers*, IEEE Computer Society Press, 1995.

Period Papers: will examine an assigned set of years (such as 1974 to 1976) in the periodical literature and identify the major activities in the computing **industry** for each year. To provide consistency, students must use *Datamation* (started in 1954) and *Computerworld* (started in 1967). A summary of the paper will be presented to the class. The use of web sites is discouraged.

Technology Project (GRADUATE STUDENTS ONLY): will examine a significant technological development, such as processors, mass storage, operating systems, computer languages, etc. Papers will trace the development from its origins to the present. Students may also trace a specific application within an organization active in using a computer before 1980. **Topics must be approved by the instructor**, and will be presented to the class.

Proposals must identify (a) the topic, (b) the methodology to be used, (c) your research schedule, (d) sources already consulted (with appropriate comments), (e) additional sources or activities planned, and (f) the relevance of the topic to the history of computing and this class.

GRADING

	U/G	Grad
Midterm Examination (take home)	40%	20%
Biography project	30%	20%
Period paper & presentation	30%	20%
Technology project	N/A	40%
- proposal	10%	due: 10/16
- draft	15%	11/20
- paper	20%	12/11
- presentation	5%	

NOTE: late materials will be penalized a letter grade for each class period that they are late! No exceptions!

REQUIRED READING

Michael Williams, *A History of Computing Technology*, Second Edition, IEEE Computer Society, 1997. Additional readings are in a binder, on reserve in Bender Library.

RECOMMENDED READINGS (professor has multiple copies)

William Aspray, et al, *Computing Before Computers*, Iowa State University Press, 1990

Paul Ceruzzi, *Reckoners, The Prehistory of the Digital Computer, from Relays to the Stored Program Concept, 1935-1945*, Greenwood Press, 1983

Clark Mollenhoff, *Atanasoff, Forgotten Father of the Computer*, Iowa State University Press, 1988

Jamie Parker Pearson, ed., *Digital At Work: Snapshots from the first thirty five years*, Digital Equipment Corporation, 1992

The **IEEE Annals of the History of Computing** is the principal scholarly publication in computing history. Everyone should spend some time exploring the **Annals**.

Note: Dr. Bergin is the Editor-in-Chief of the Annals.

What constitutes history is relative! This is nowhere more obvious than when one looks at old issues of *Datamation*, *Infosystems*, *Computerworld*, or other periodicals and journals. Although there are articles labeled "history of computing" which would be of interest to a student in this class, there are also articles of historical significance, i.e., written long enough ago to provide insight into what things were like at that time:

Daniel D. McCracken, "A Progress Report on Machine Intelligence", *Datamation*, October 1960.

Indeed, almost everything in a 1960s or 1970s periodical has historical significance. Moreover, the advertising of the period provides as much insight into what was going on at the time, as do reports about "new" hardware, software, and applications!

NOTE: *students are strongly encouraged to view the PBS video series: **The Machine That Changed the World*** (available in Bender Library); some parts will be used to supplement class lectures. Additional videos are identified on the Sloan web site.

WRITING QUALITY

Written materials will be judged with respect to writing quality as well as technical accuracy. Papers are expected to meet or exceed accepted undergraduate writing standards.

Questions on the use of outside materials should be referred to the instructor. Students are encouraged to acquire and use a "writer's guide" during the course, such as Turabian's *A Manual for Writers of Term Papers, Theses, and Dissertations*.

PLAGIARISM

Plagiarism is defined as taking the language, ideas, or thoughts of another, and representing them as your own. If you use someone's ideas, cite them; if you use someone's words, clearly mark them as a quotation. Plagiarism includes using another's computer programs and procedures. All instances of plagiarism will be reported to the Dean of the College of Arts and Sciences for appropriate action.

CLASS SCHEDULE

1. Introduction and 'Numeration'

An explanation of the purpose and focus of the course, discussion of the syllabus and requirements; the concepts of numbers and numeration.

Reading: Williams: Preface, 1

2. Preconditions for Computing

A discussion of the major counting, computing, and record keeping antecedents of modern computers: abacus, logarithms, slide rule, and Napier's bones as well as the efforts of Schickard, Pascal, and Leibnitz.

Reading: Williams: 2 & 3 to 3.5

Gardner: "The Soul of an Old Machine," *Discover*, May 1985.

Lab: creating and using the slide rule

3. Charles Babbage and Joseph Jacquard

Examination of the life and machines of Charles Babbage, and the development of automated weaving looms.

Readings: Williams: 4

4. Herman Hollerith and Early Accounting Machines

An examination of the development of punched cards, electronic accounting machines and the start of an industry.

Readings: Biles, et al, "Herman Hollerith: Inventor, Manager, Entrepreneur -A Centennial Remembrance," *Journal of Management*, Vol. 15, No. 4, 1989
Keith Reid-Green, "The History of Census Tabulation," *Scientific American*, February 1989.

Lab: interpreting punched cards

5. Information and Office Appliances (the Mechanical Office)

An examination of the evolution of commercial adding machines and typewriters, as well as their information-bearing cousins: the teletype, telephone and the wireless!

Lab: early adding, calculating, and typing devices.

Video: (if time available) The Telephone (PBS)

Reading: Williams, Chapter 3.7

6. Analog Devices and Mechanical Monsters

An examination of the work of Stibitz, Zuse, Aiken, and IBM. A discussion of early efforts to use mechanical devices.

Reading: Williams: 5, 6; Slater: Zuse, Aiken

Video: *Computer Pioneers and Pioneer Computers*, TCM#1

7. Electronic Revolution, Atanasoff and ENIAC

A discussion of the work of John Atanasoff at Iowa State, and John Mauchly and J. Presper Eckert at the University of Pennsylvania.

Reading: Williams: 7

Video: *The Machine that Changed the World: "Giant Brains"*

Take home examination handed out!

8. Early Stored Program Computers: EDVAC, EDSAC, Whirlwind, IAS, the Defense Calculator (IBM 650)

Evolution of the stored program concept and the computers of the *zeroth* generation.

Reading: Williams: 8

Video: *Computer Pioneers and Pioneer Computers*, TCM#2

Take home examination DUE

9. Mainframe Computers: Birth of an Industry

Development of the commercial computer industry, early machines and the early entrepreneurs.

Readings: Williams, Chapters 8 & 9

Video: *Machine that....: "Inventing the Future"*

Biography papers DUE

10. The Transistor and Mini-computers

This lecture will focus on the growth of the Digital Equipment Corporation and the PDP line of minicomputers as an example of the minicomputer era.

Readings: Pearson, *Digital At Work*

Video: *Transistorized*, (PBS, Digital Equipment Corporation)

11. The Computer Industry, Year by Year

Class members will present their period reports, identifying major contributions and contributors to the growth of the computer industry. (Estimated time: 10 minutes each)

NOTE: Period Reports are DUE

12. The Evolution of Programming Languages

Guest: Jean Sammet, author of: *Programming Languages: History and Fundamentals*, Prentice-Hall, 1969

Handout: Sammet, "Some Approaches to, and illustrations of, Programming Language History, *Annals of the History of Computing* (reprinted by permission)

Reading: Slater: Hopper, Backus, Kemeny and Kurtz, Ritchie
And Thompson

13. Microcomputing and Personal Computing

Video: *Triumph of the Nerds: An Irreverent History of the PC Industry*, PBS.

14. Graduate Research Paper Presentations

NOTE: RESEARCH PAPERS ARE DUE (paper and diskette)

Supplemental Reading List

Herman H. Goldstine, *The Computer, from Pascal to von Neumann*, Princeton University Press, 1972. (2nd ed. 1996).

N. Metropolis, J. Howlett, Gian-Carlo Rota, *A History of Computing in the Twentieth Century*, Academic Press, 1980.

Maurice Wilkes, *Memoirs of a Computer Pioneer*, The MIT Press, 1985.

Nancy Stern, *From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers*, Digital Press, 1981.

Kent C. Redmond, Thomas M. Smith, *Project Whirlwind, The History of a Pioneer Computer*, Digital Press, 1980.

Charles Bashe, Lyle Johnson, John Palmer, Emerson Pugh, *IBM's Early Computers*, MIT Press, 1986.

Emerson W. Pugh, *Memories That Shaped an Industry, Decisions Leading to IBM System/360*, The MIT Press, 1984.

C. Gordon Bell, J. Craig Mudge, John E. McNamara, *Computer Engineering: A DEC View of Hardware Systems Design*, Digital Press, 1978.

Tracy Kidder, *The Soul of a New Machine*, Atlantic-Little Brown, 1981.

Thomas Bergin and Richard Gibson, *History of Programming Languages*, ACM Press, Addison Wesley, 1996.

Richard L. Wexelblat, *History of Programming Languages*, Academic Press, 1981.

Susan Lammers, *Programmers at Work*, Microsoft Press, 1986.

Freiberger and Swaine, *Fire in the Valley: The Making of the Personal Computer*, Osborne/McGraw-Hill, 1984.


James L. McKenney, *Waves of Change, Business Evolution through Information Technology*, Harvard Business School Press, 1995.

J. David Bolter, *Turing's Man, Western Culture in the Computer Age*, University of North Carolina Press, 1984.


Appendix B: Sample Pages from a Class Handout (PowerPoint)

Figure 5. Sample Handout from the Class (1)

Charles Babbage (1791-1871)



Never to be completed



- December 1830, a dispute with his chief engineer, Joseph Clement, over control of the project, ends work on the difference engine
- Clement is allowed to keep all tools and drawings by English law


Charles Babbage (1791-1871)

- Born: December 26, 1791
- son of Benjamin Babbage a London banker [part of the emerging middle class: property, education, wealth, and status]
- Trinity College, Cambridge [MA, 1817] with John Herschel and George Peacock, produced a translation of LaCroix's calculus 1802

Importance of the Difference Engine

- 1. First attempt to devise a computing machine that was automatic in action and well adapted, by its printing mechanism, to a mathematical task of considerable importance.
- 2. An example of government subsidization of innovation and technology development
- 3. Spin off to the machine-tool "industry"

A vision of calculating by steam!

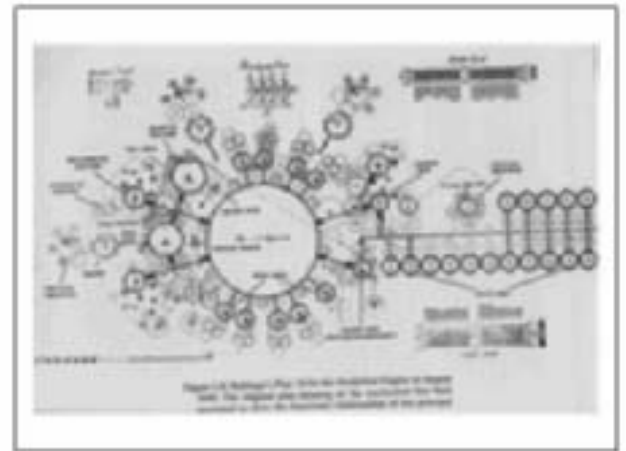


My friend Herschel, calling upon me, brought with him the calculations of the computers, and we commenced the tedious process of verification. After a time many discrepancies occurred, and at one point these discrepancies were so numerous that I exclaimed 'I wish to God these calculations had been executed by steam.'
1837

Science Museum's Reconstruction

- *Difference Engine Number 2* [1847 to 1849] constructed according to Babbage's original drawings [minor modifications]
- 1991 Bicentenary Celebration
- 4,000 parts
- 7 feet high, 11 feet long, 18 inches deep
- 500,000 pounds

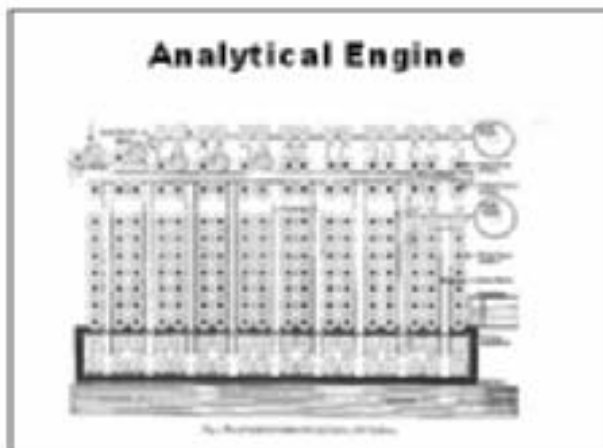
Figure 6. Sample Handout from the Class (II)



Ada Augusta Byron, 1815-1852



- born on 10 December 1815.
- named after Byron's half sister, Augusta, who had been his mistress.
- After Byron had left for the Continent with a parting shot -- "When shall we three meet again?" -- Ada was brought up by her mother.



**Ada Augusta Byron,
Countess of Lovelace**

- Translated Menabrea's paper into English
- Taylor's: "The editorial notes are by the translator, the Countess of Lovelace."
- Footnotes enhance the text and provide examples of how the Analytical Engine could be used, i.e., how it would be programmed to solve problems!
- Myth: "world's first programmer"

Appendix C: Sample Midterm Exam

Spring 2001

Take Home Midterm Examination: CSIS 550 History of Computing

Due: March 8, 2001

Think of this exam as a set of research projects. For some questions, the materials in the textbook and lectures will be adequate; for other questions, *additional research will be necessary!* Such research should not be solely from the World Wide Web but should use printed sources such as books and periodicals such as the *Annals of the History of Computing*.

Please note that additional material is expected of **graduate students** and that the use of such materials should be properly referenced.

1. Read an article in *The Annals of the History of Computing* and prepare an article review identifying: (a) the focus of the article, (b) the sources used by the author, (c) what you learned from the article, and (d) how it relates to the course.
2. Trace the history of the efforts to improve arithmetic processes. Be sure to identify at least **four early aids to calculation** (prior to 1800), and explain how they functioned.
3. Who was **Charles Babbage**? Describe his contributions to the evolution of computing. Who was **Ada Augusta Byron** and for what is she known? What kept Babbage from realizing his vision?
4. Who was **Herman Hollerith**? Describe his invention, the context in which it took place, and the need which it fulfilled. Relate the evolution of Hollerith's machinery to the data processing machinery of the 1930s, being sure to identify the types of processing performed at that time.
5. Identify **three information appliances** used in the late 1800s to simplify and expedite office processes.
6. Describe the calculating machines designed by **Konrad Zuse, George Stibitz and Howard Aiken**. What elements of their designs can we find in modern computers?
7. Who was **John Vincent Atanasoff**? Describe his motivation and effort to build a computer. What insights did Atanasoff have that are still being used in computers today?
8. Describe the work of **J. Presper Eckert and John Mauchly** at the University of Pennsylvania. How did this machine differ from earlier machines?
9. What is a computer? What do you consider to be the "**first computer?**" *Justify your answer by reference to early computing projects, and cite the sources for your argument.*
10. Identify the subject of your **biography project**. What sources of information have you identified to use in this research project? What have you learned so far?

Graduate students: a written proposal on the technology project is due on March 23rd.

The proposal must contain the following: (1) topic/subject/title, (2) a general description of your research or a thesis statement (as appropriate) (3) sources of information identified so far, (4) rationale for the research (why it is interesting or important), (5) how it relates to the class.

Students are strongly encouraged to make an appointment to discuss their project with the instructor after semester break! **The research papers will be presented to the class on April 27th and May 4th!**

All students: The Biography paper is due on March 23rd.

The Period papers are due on April 5, and presentations will begin on that date, and will be continued on April 12th.

11. Enhancing Learning Using the History of Computing: Opportunities to Use History in an Introductory Computers and Information Course

Thomas J. Bergin
American University

Abstract

Although the author believes that there are opportunities in all computer courses to use history to enhance student learning, the use of history and artifacts is an effective strategy in helping students understand the complexities of modern computer hardware and software. This paper describes how the author used computer history and computer artifacts to help students understand memory, storage media, and other topics in an introductory course on computers and information systems for non CS/IS majors. This paper is a companion to “Resources for a Course on the History of Computing” and “History of Computing for Computer Science and Information Systems Students,” both in this volume. A syllabus for a one-semester Introduction to Computers and Information course is included.

Before beginning, I think it only fair to relate some critical insights gained in my early career, which I attempted to leverage in the classroom. I finished my first year of graduate study in English and Comparative Literature at the University of Maryland in May 1966. Since I had just become engaged, a friend suggested that I look into working with computers at the Veterans Administration in Washington, D.C. After passing an (IBM-provided) aptitude test, I began my career the day after Memorial Day.

As a newly hired *digital* computer systems analyst (trainee), I struggled through three courses offered at the IBM Education Center in Washington, D.C.¹⁰⁹ As I recall, the first course was titled, “What is a Computer?” Because computers were fairly simple in those days, I was able to understand the material and thought I was ready for more. The second course, “Assembly Language for the IBM System 360,” was immediately followed by a course in “Operating Systems Principles.” Needless to say, since I had majored in English, I was completely lost. When I told my supervisor that nothing made sense to me, he said, “Stick it out. It is hard to get slots at the Education Center these days!” Needless to say, I read whatever I could find over the summer and enrolled in a graduate course, “The Systems Approach,” at American University in September 1966.

Aware of my enrollment in a graduate course, I was pressed into service as a lecturer in a week-long training program for computer users when my supervisor was on vacation. The purpose of the “VA Management ADP Institute” was to provide a fundamental understanding of how a computer worked, how to program a computer, and overviews of the various applications under development in 1966.¹¹⁰ The audience was made up of supervisors in the areas being automated. Some of the early computer projects at the Veterans Administration were: automated payroll processing, various accounting applications, engineering administration, and some rudimentary aspects of medical administration. My early projects were in engineering administration.

¹⁰⁹ As I recall, so few people knew anything about computers in those days that most organizations were forced to hire “trainable people” and send them to various schools offered by the mainframe vendors.

¹¹⁰ ADP stood for Automatic Data Processing, a term that encompassed the use of punch card accounting machines. The term was replaced with EDP (Electronic Data Processing) when computers started to proliferate in large organizations.

The first “module” that I delivered was “How a Computer Works.” This covered the parts of a computer system: input devices, central processing unit (CPU), secondary memory, and output devices. As part of this lecture, I showed the students photographs of the IBM 1401 computer, including the card reader, the central processing unit, the tape drives, and the printer. After giving them copies of the standard systems chart, I explained how each component worked. Input in those days was usually punched cards, so I explained the numeric and zone punches and how they encoded numbers, alphabetic characters, and special characters. After this discussion, I passed out a card with a short sentence in it and a card with all the characters punched. Students then spent ten or fifteen minutes decoding the card. I told them that they would have a chance to punch a card when we visited the Data Processing Center (DPC) at the end of the day. This exercise and explanation helped to demystify the storing of information (on inert media).

When the parts of the CPU were discussed, I explained *how memory* worked, *how data* was stored in the memory, and *how calculations* were executed (as part of a *computer program*). The ADP Institute had a teaching aid, in which seven electric lights (with 60-watt bulbs) were mounted on a large frame that hung over a portable blackboard. With this device, I could explain how the “*bits*” of a character were encoded as “*ons*” and “*offs*” and how the parity bit worked.¹¹¹ Once students understood this, they could see how the punched card code was mapped into Binary Coded Decimal (BCD) code.¹¹² Later, when I gave lectures at VA field offices, I had a small box made that used lights from our family Christmas tree. This simple classroom tool has never failed to work: the bulbs on the device have worked perfectly for over thirty years, and, more importantly, the “lights” always came on for the students too! A most effective gadget!

The IBM 1401 had a “core memory,” where small *cores* of ferro-magnetic metal were strung on wire frames. After explaining how the core was magnetized (to indicate “on” or a “1”), I drew a diagram on the blackboard, showing how a (BCD) character would look. Then I held up a core plane for everyone to see and passed it around. Students saw the core plane as a physical piece of equipment and accepted the concept of storing data in a computer memory. Although *memory* was a little more sophisticated than the punched card (hole or no-hole), the idea of cores representing “on” or “off” seemed acceptable to most students.

The *piece de resistance* of the “show-and-tell” was a reel of *200 bpi magnetic tape*. After an explanation about how “the tape head” (which is composed of electromagnets) magnetizes tiny spots on the tape, the concept of data encoding and storage was reinforced by having each student *develop a piece of magnetic tape!* At that time, 200 bpi tape encoded 200 characters in one inch of tape. This low density allowed us to use something called *Magna-See*, to actually see the magnetic spots on the tape.¹¹³ I cut short pieces of tape for each student and they dipped their tapes in the *Magna-See* solution. After the fluid evaporated, tiny iron filings

¹¹¹ Parity was a strategy to check characters to see that no bits were dropped. In even parity, a character must have an even number of bits; if a character was encoded with an odd number of bits, the parity bit was turned on to make the total number of bits even. If the computer encountered an odd number of bits, an error message was sent to the operator.

¹¹² Binary Coded Decimal code used 7 bits to encode each character: 4 numeric bits, 2 zone bits and a parity bit. With the advent of the IBM System 360, a 9-bit code was used called Extended Binary Coded Decimal Interchange Code (EBCDIC), which had 8 data bits and parity.

¹¹³ Magna-See™ was used to check the tracking of magnetic tape in the recording (radio, television, phonograph, etc.) industry, and was carried over to computing as a way of seeing if the read-write head on a magnetic tape was aligned properly. I believe it was still used in the 1990s in the recording industry, since a colleague (who had used all of my supply) was able to get us a partial can.

remained where the tape head had magnetized the spots. We then put the tape on an index card and covered it with clear plastic tape, creating a keepsake. I put the first one into a clear glass paperweight, and the encoding is still visible 38 years later.

After the lecture, the class took a tour of the Data Processing Center in the basement of the building and saw the various machines discussed in the lecture. Operators showed the students decks of punched cards, reels of paper and magnetic tapes and assorted disk packs, and demonstrated many of the things we had talked about in class (i.e., *reading* cards, *writing* tapes, and *printing* output). In the programming segment, all students programmed the “Pilgrim’s Dilemma.” As part of this class, students wrote 1401 SPS (symbolic programming system) programs to calculate the amount of money Peter Minuit could have earned if he had invested the \$24 he spent for Manhattan Island in an account at 6 percent interest. Since this transaction occurred in 1626, the interest is astounding and the power of compound interest is rarely forgotten by anyone who does the arithmetic.¹¹⁴

Why am I telling you this? I am sharing these experiences because they influenced my approach to teaching and, ultimately, my career. The students at the ADP Institute were never going to be “computer people.” They only wanted to *understand* how this machine worked, that is, how metal and plastic did the work that their subordinates (and earlier generations of machines) were doing. More importantly, only if these supervisors and managers *understood* how this machine worked would they be comfortable with *automation*, and, moreover, have the confidence to believe the data and reports produced by the computers. The bottom line was that if we did not succeed in teaching these people about computers, our applications would not be accepted and all our efforts would be in vain.

A second reason to share these experiences is because the use of artifacts and demonstration pieces made sense to me and helped me understand some rather arcane concepts. At the VA ADP Institute, it was easy to see the very positive effects of students seeing and interacting with computer materials. Seeing the *spots* appear on the tape took the “magic” out of what was a very new and confusing subject in the early days of computing.

American University

In 1982, I left the Federal Government and embarked on a teaching career. One of my first tasks was to design a new Introduction to Computers course for graduate students majoring in information systems. After a quick review of available textbooks, I came across a book by Capron and Williams that I thought would do the trick.¹¹⁵ The text consisted of three parts: how a computer worked, how to program a computer, and some chapters on systems analysis (and tools), security of records, and so on. The first chapter was a brief history of computing—a fun place for me to start.

It is interesting that over the years, as new subjects needed to be integrated into thousands of Introduction to Computers (Computing) textbooks, the chapter on the history of computing was moved to an appendix by a few authors and dropped by numerous others.¹¹⁶

¹¹⁴ The number had 16 zeroes in it as I recall, and I carried this problem over to my university teaching using BASIC, Pascal, and Visual Basic over the years.

¹¹⁵ Unfortunately, I no longer have a copy of this book, so I cannot verify any of the facts of publication.

¹¹⁶ The last version I used in the classroom is H.L. Capron, *Computers, Tools for an Information Age*, sixth edition (Boston, 2000), which contained a nice appendix on the history of computing.

Indeed, it is important to note (for those not familiar with “intro” texts) that most publishers provide instructors with a wide range of classroom support including PowerPoint® slides, quiz/exam generators, Instructor’s Guides, and supporting websites. All of these tools make teaching such a course significantly easier.

In the fall of 1998, I received a grant from the Alfred E. Sloan Foundation as part of their *Science and Technology in the Making (STIM)* project. The purpose of the *STIM* project was to capture the history of various technologies that Sloan believed would be ignored by historians because science and technology were expanding so rapidly. The *STIM* projects were experiments to see if people’s experiences with various new technologies could be documented using web-based tools. Our research team created a web site to capture the histories of: 1) programming languages, and 2) software engineering: www.computinghistorymuseum.org/. In addition, the website contained photographs of artifacts in the *Computing History Museum*, which we hoped would bring people to the site and, thus, further the project goals.¹¹⁷ The first page of the site is shown in Figure 1.

In December 2000, as part of a request for a no-cost extension, we requested approval to enlarge the site by adding two additional sections: the first to support *teaching the history of computing*, and the second to support the Editorial Board of the *IEEE Annals of the History of Computing*.¹¹⁸ I had been teaching a course in the history of computing since 1988 and wanted to make the syllabus and lecture notes (in the form of PowerPoint® slides) available to others in the hopes of helping them teach this important subject. The teaching page is shown in Figure 2.

Because I continued to teach introductory courses until I retired, I created a ninety-minute lecture on the history of computing suitable for use in introductory courses. The Overview was distilled from the lectures for the History of Computing course, discussed in the companion paper, “History of Computing for Computer Science and Information Systems Students.” Another companion paper, “Resources for a Course on the History of Computing” provides a longer discussion of the Computing History Museum, with specific attention to the Teaching pages and the fourteen PowerPoint® lectures.

¹¹⁷The Computing History Museum occupied two rooms in the foyer of Clark Hall on the American University campus. Unfortunately, the Museum no longer exists. After Dr. Bergin retired in 2002, the University renovated Clark Hall for the School of International Service. A virtual tour of artifacts is still available on the CHM website and may be useful to faculty and students interested in the history of computing.

¹¹⁸The IEEE Annals of the History of Computing is the journal of record for this subject. Dr. Bergin served as the Editor-in-Chief from 1999 to 2003 and as Assistant Editor-in-Chief from 1996 to 1999.

Figure 1. Computing History Museum Home Page



Figure 2. Teaching the History of Computing Page



Lecture 0: Overview of the History of Computing

This lecture contains material on the abacus; Charles Babbage, the Difference Engine, the Analytical Engine, and Ada Augusta Byron; Herman Hollerith and the evolution of punched card processing; the origins of IBM; and the Electrical Numerical Integrator and Computer (ENIAC). In addition, there are slides on “stored programming”; some early British computer projects; the evolution of “mainframe” (first-, second-, third-, and fourth-generation) computers; the mini-computer industry; and a few slides on micro-computers and the early Intel® microprocessors.

The “Overview of the History of Computing” lecture is shown in Figure 3.

Figure 3. Lectures in the History of Computing



The screenshot shows the website for the American University Computing History Museum. The header includes the museum's name and navigation links: Home, Museum, Teaching the history of computing, Class project, and IEEE Annals. A sidebar on the left lists navigation options: Introduction, Syllabus, Lectures, Class Materials, Student Projects, Discussion, Bibliography, and Resources. The main content area is titled "Lectures in the History of Computing" and contains a paragraph explaining that the page features Professor Tim Bergin's lectures from the class CSIS 550. Below this, two lecture entries are listed:


- Overview of the History of Computing**
 - History of Computing Overview ([View](#) | [Download](#))
- Lecture 1**
 - Numeration ([View](#) | [Download](#))

As shown in Figure 3, the lectures can be *viewed* or *downloaded*. In this way, faculty can look at a lecture to see if it meets their needs, and then download and modify it for content and personal style. In the past two years, a number of instructors from other universities have requested permission to use the slides in their classes. Although permission to use the materials is not required, we do ask that they use “normal academic courtesies” with respect to citing the Computer History Museum as the source of the materials, and that they provide us with feedback so that we might correct mistakes or add materials in the future. It is important to make some observations at this point: first, the reader should log on to the website in order to make sense of this list of topics: <http://computinghistorymuseum.american.edu/>. In this way,


you can view the entire lecture and gain some sense of how it might fit your requirements. A second observation is equally practical: a valuable feature of using PowerPoint@s that the lecture slides can be printed (in black and white) as student handouts. Figure 4 is a sample handout from the lecture on Charles Babbage.

Figure 4. Sample of Class Handout

Charles Babbage (1791-1871)



Never to be completed



- December 1830, a dispute with his chief engineer, Joseph Clement, over control of the project, ends work on the difference engine
- Clement is allowed to keep all tools and drawings by English law


Charles Babbage (1791-1871)

- Born: December 26, 1791
- son of Benjamin Babbage a London banker [part of the emerging middle class: property, education, wealth, and status]
- Trinity College, Cambridge [MA, 1817] with John Herschel and George Peacock, produced a translation of LaCroix's calculus 202

Importance of the Difference Engine

1. First attempt to devise a computing machine that was automatic in action and well adapted, by its printing mechanism, to a mathematical task of considerable importance.
2. An example of government subsidization of innovation and technology development
3. Spin off to the machine-tool "industry"

A vision of calculating by steam!



My friend Herschel, calling upon me, brought with him the calculations of the computers, and we commenced the tedious process of verification. After a time many discrepancies occurred, and at one point these discrepancies were so numerous that I exclaimed 'I wish to God these calculations had been executed by steam.'
1832

Science Museum's Reconstruction

- *Difference Engine Number 2* (1847 to 1849) constructed according to Babbage's original drawings [minor modifications]
- 1991 Bicentenary Celebration
- 4,000 parts
- 7 feet high, 11 feet long, 18 inches deep
- 500,000 pounds

A third point is that the lectures contain more material about micro- and personal computers, and it should be easy for instructors to download lectures and add additional slides to the Overview lecture. And lastly, the Computing History Museum website offers a Virtual Tour of the exhibits, which might prove useful to instructors or students. Two examples of artifacts in the Museum are shown in figures 5 and 6.

Figure 5. Chinese Swan Pan (Abacus)



Figure 6. Schickard Calculator (Re-creation)



Discussions of artifact-collecting, the origins of the Computing History Museum at American University, and integrating history into courses are provided in the companion papers,

“History of Computing for Computer Science and Information Systems Students” (chapter 10) and “Resources for a Course on the History of Computing” (chapter 27).

Introduction to Computers and Information Course

As mentioned earlier, the content of the Introduction to Computers and Information course has changed dramatically from the early 1980s to the present. Twenty years ago we focused on how the machine operated, how to program it to do useful work, the systems analysis process—used to create computer applications—and a few management of computing issues such as security and privacy. The introduction and widespread use of personal computers in the mid-1980s has pushed much of that material to the margins of such courses, and significant amounts of time are now spent teaching students to use word processors, spreadsheets, database packages, and the tools of the Internet and World Wide Web.

This applications-oriented material now dominates such courses, but students are still better off if we can give them some insight into *how a computer works* and *how sophisticated applications programs are created*. Although many readers will think that this is unimportant, I believe that we can create better computer users if we can take the mystery out of personal computing. Indeed, the use of artifacts in the classroom, “*show-and-tell*” if you will, is a powerful teaching tool that makes this sometimes invisible and obscure technology real, as mentioned in the early parts of this paper.

As stated above, the textbook publishers provide exam-generation software with questions keyed to the textbook. Using one of these, you can create a multiple choice exam in an hour.¹¹⁹ One of the things that I do is put a Part II on the Midterm Exam (after we have covered the hardware material) that states:

You have a job over the semester break, and in conversation you mentioned that you had this terrific computer class last semester! Your boss wants to purchase a new computer for his home and he wants you to explain the advertisement on the previous page. Your boss isn't the brightest bulb in the pack, so be clear, concise and thorough.

(Note: this question is worth 20 points)

Most students can explain the ever-present advertisements in the newspapers, but do they really understand the jargon? Can they explain it to someone else's satisfaction (like the parents paying upwards of \$100,000 for four years of college)? A recent advertisement for a Dell computer (www.dellforme.com) advertised a computer with:

- *an Intel® Pentium® 4 processor with HT technology*
- *(operating at) 3.40 GHz*
- *(using an) 800 MHz front side bus and*
- *an Intel® 875P chipset with ICH5R.*

Another advertisement offered:

- *FREE UPGRADE! 384MB, 266MHz, 2DIMM*
- *(from 256MB, 266MHz, 2DIMM)*

¹¹⁹ Such textbooks also have other question styles such as fill-in-the-blanks, true/false, and short essay-type questions.

Do instructors really explain what all that means—to the satisfaction of any student? Do any of us really believe that students in an introductory class can understand what it means to operate at 3.4 billion cycles per second? Can we put into practical terms how much better the 80 GB (7200 RPM) hard drive is than the 40 GB model?

If students can learn how older, simpler technologies like a “core” memory work, then the newer technologies with their extraordinary capacities may become understandable to people other than computer science and computer engineering majors. Thus, the demonstrations referred to in the early part of this paper and the use of artifacts may help instructors in intro classes to bridge the modern techno-babble gap.

For the hardware classes, I drew upon my collection of artifacts, but suffice it to say that the older the better. Students liked seeing racks of vacuum tubes from first-generation mainframe computers and plug modules from minicomputers like the PDP-8. Indeed, when I dragged out old personal computers, like the Commodore 64, some student always said, “That was my first computer!” Students also liked seeing (and touching) 10” floppy disks (from a TRS-80 Model II, 1979) and 5 ¼” floppy disks (from an Apple II, 1977). There was always a chuckle when they heard that the Texas Instruments 99-4A, which I bought in 1981 at Toys R Us for \$299 (less a \$100 rebate), used a *cassette tape* for storage.¹²⁰ I have come to understand that “modern” people have little or no understanding of history, even the history of *twenty years ago*. Indeed, it is sometimes difficult to remember that most of our freshmen weren’t even born twenty years ago! Indeed, the early days of personal computing occurred before they were born, as did the Pilgrims and the dinosaurs!

Other artifacts appropriate to classroom use may be found at garage sales, on the web and at computer swap meets. Although it takes time to look for such things, they are invaluable in the classroom. You might also share your experiences with students and see how they react. Imagine my telling them about my second computer: an IBM PC, purchased in 1983, which had:

- 64 K of memory,
- an Intel 8088 processor operating at 4.77 MHz,
- two 5 ¼” floppy drives,
- color monitor with four colors,
- a “dot-matrix” printer,
- a modem that operated at 4,800 bits per second,
- (no software), only PC-DOS Version 1.0 and PC-BASIC,

and cost \$4,900 at a computer store that had recently opened in Frederick, Maryland.¹²¹

Some artifacts that may still be found at swap meets and antique shops are: magnetic tapes, a variety of hard and floppy disks (including: 10” floppies, 5 ¼” floppies, 3 ½” diskettes), and modules from mini- and mainframe computers. A lot of places also have stacks of old

¹²⁰ The TI-994 was introduced in 1980. I bought mine to do word processing for my first book; the rebate was offered on August 6, 1983; the final price in late 1983 was \$49.95.

¹²¹ And was used for James Lefter and Thomas J. Bergin, *A Microcomputer-Based Primer on Structural Behavior*, Book disk package (New York, 1986). The diskette was formatted for IBM on one side and contained programs in PC BASIC and Lotus 1-2-3 on the other side. It was formatted for Apple BASIC and VisiCalc. Jim Lefter and I think this might be the first “flippy.”

micro-computers and personal computers. One of my favorite show-and-tell items is the 10' floppy diskettes for the TRS 80 Model II, which became available in 1979.¹²²

Conclusion

It is easy to think that students understand us. After all, we try to think and speak clearly, don't we? The problem of teaching an Introduction to Computing course is that students accept the modern world at face value and use computer jargon with ease. But do they truly understand what all the techno-words mean? And more importantly, what can college instructors do about the constant and rapid changes in the computing and allied technologies that surround us and that we all take for granted?

When I first went to American University, I was on the faculty of the Center for Technology and Administration with one Professor Richard Bassler. Dick was a crusty sort, with many years of experience in the early days of computing. He used to call anyone who did not understand how a personal computer worked a "toaster-head" because they operated the computer like it was a toaster! They put one or two disks into the slots, typed in some data, and out popped the answer! One day, I remember replying, "People don't know how a toaster works, but they can still make toast!" Those were the good old days.

Thomas J. (Tim) Bergin is Emeritus Professor of Computer Science at the American University in Washington, D.C. Dr. Bergin entered the computing field in 1966, as a "digital computer systems analyst (trainee)" with the U.S. Veterans Administration. In 1980, he received an Inter-governmental Personnel Act Fellowship as a Visiting Professor in American University's Center for Technology and Administration. In 1982, he joined AU as an Assistant Professor and Director of the Quantitative Teaching and Research Laboratory, and managed the introduction of personal computers to support faculty and student research. He has authored or edited four books: *History of Programming Languages* (ACM Press, 1996) and *Fifty Years of Army Computing: From ENIAC to MSRC* (Army Research Laboratory Special Report-93, September 2000), *A Microcomputer-Based Primer on Structural Behavior* (with James Lefter, Prentice-Hall, 1986) and *Computer-Aided Software Engineering* (Idea Group Publishing, 1993). In addition to his teaching duties, Dr. Bergin created the Computing History Museum at American University, which was renovated out of existence upon his retirement in 2002. To see some of the artifacts mentioned in this article you can visit the Virtual Computing History Museum (<http://computinghistorymuseum.american.edu/>). A story about the creation of the Museum was also featured in an article in *Yesterday's Office*.¹²³ Dr. Bergin also served as the Assistant Editor-in-Chief of the *IEEE Annals of the History of Computing* from 1995 to 1999 and as Editor-in-Chief from 1999 to 2003. Dr. Bergin also served as historian to the ACM for their 50th anniversary celebration. *Author's address:* 217 Pewter Lane, Silver Spring, MD 20905 USA. Email: tbergin@american.edu.

¹²² All dates and specifications are from Thomas F. Haddock, *A Collector's Guide to Personal Computers and Pocket Calculators: A Historical, Rarity and Value Guide* (Books Americana, 1993).

¹²³ <http://www.yesterdaysoffice.com/index.cfm?fuseaction=ShowArticle&articleid=36>

Appendix A. Course Syllabus

Figure 7. *American Magazine* (Winter 1982)



Artist: Dan Sherbo.

Source: *American Magazine* of The American University (Winter 1982): 11.

**The American University
College of Arts and Sciences
Department of Computer Science and Information Systems**

Fall 2000

CSIS-100.01 Computers & Information

Tuesdays and Fridays: 8:30 to 9:55 in Ward 101

**Thomas J. Bergin, Ph.D.
Clark Hall 118 885-3863
*tbergin@american.edu***

Office Hours: Tue. & Fri. 10 to 12 and 1 to 2 (and by appointment)

DESCRIPTION AND PURPOSE

This course is intended as the **first course** for students with no formal computer course work or training in computing. The course introduces students to a wide variety of computer-based tools and resources such as spreadsheets, database management systems, presentation packages, the Internet and the World Wide Web. Topics include software, hardware, systems development, societal issues, and future directions.

COURSE OBJECTIVES:

1. Provide students with an introductory level course in computer concepts, hardware, software, terminology, etc.
2. Provide an introduction to personal computer software tools: Microsoft Windows, Excel, PowerPoint, and Access.
3. Become familiar with Internet applications such as electronic mail, and the World Wide Web.
4. To provide some insight into the direction and continuing advances in computers and information technology, and their impacts on our daily lives.

METHOD OF INSTRUCTION:

The format to the course is primarily that of lectures, class discussions, and workshop sessions. Since the class will hold a number of sessions in the Anderson Laboratories, students will gain hands-on experience in using computers.

COURSE REQUIREMENTS:

Article Reviews

Articles about computers and computing appear in all types of periodicals, from popular magazines to the scholarly journals for almost every field of knowledge. Article Reviews are intended to introduce students to this rich literature.

Article Reviews should be 1 page in length, and discuss: **(1) the major ideas** in the article, **(2) how the article relates** to the course, and most importantly, **(3) your thoughts**, reactions and criticisms of the author's ideas. **All three parts must be clearly identified or points will be deducted (hint: use headings!!!).**

Students are required to do one article review from three literatures: **(1) popular** literature (*Time*, *Business Week*, etc.), **(2) your major area of study** (*Public Administration Review*, *Journal of Sociology*, etc.) and **(3) the literature of the computer industry** (*ComputerWorld*, *PCWorld*, *MacWorld*, etc.)

PC Software Projects

Each student will use a number of software packages during the course. For each type of software, students are expected to plan, execute, and document a project of their choice. For Access, we will create project teams. Students are expected to identify problems that are meaningful to them, and to document their work thoroughly. Projects may be based on materials

from the class textbook or other materials containing suitable data; questions should be discussed with the instructor.

REQUIRED TEXTS:

H.L. Capron, *Computers: Tools for an Information Age*, Sixth Edition, Prentice-Hall, Inc. (2000)

Note 1: *In order to maximize learning, students are expected to read the assignments before coming to class.*

Note 2: *The classes on computer applications will be conducted as workshops and cover the basics of each application; students are expected to increase their proficiency, on their own, through the class projects.*

GRADING:

Article reviews	10 points
Mid-term examination	25 points
PC software projects:	60 points, as follows:
World Wide Web Research	10 points
Web Page	10 points
PowerPoint Project	10 points
PowerPoint Presentation to the Class	5 points
Excel Project	10 points
Access Project (Team project)	15 points
Class Participation	5 points

An *extra credit project* may be submitted which can count for up to 10 points. Students are to visit the Information Age exhibit at the Smithsonian's Museum of American History and write a 4 to 6 page paper on the exhibit and what they learned by viewing it.

"A" indicates achievement of distinction. It involves conspicuous excellence in all or most aspects of course expectations. "A-, B+, and B" grades indicate performance which exceeds expectations. These grades are achieved by excellence in some part of the course requirements. Grades of "B-, C+, and C" indicate that the basic requirements of the course have been met. Grades of "C- and D" are given for work which falls below acceptable standards. Assignments are expected on the due date; late assignments will be reduced by a letter grade for each class period the assignment is late!

WRITING QUALITY:

Written materials will be judged with respect to writing quality as well as technical accuracy. All submissions are expected to meet or exceed accepted undergraduate English and scholarship standards.

PLAGIARISM:

It should be noted that plagiarism is taking the language, ideas, or thoughts of another, and representing them as your own. If you use someone's ideas, cite them; if you use someone's words, clearly mark them as a quotation. Plagiarism includes using computer programs and procedures written by others.

Instances of plagiarism will be reported to the Dean of the College of Arts and Sciences.

CLASS SCHEDULE

Date	Lab	Topic	Reading
8/29		Introduction to the course and requirements	
9/1		Computer Hardware: Meeting the Machine <i>Photo Essay: the Age of Information</i> <i>Gallery 1: Making Microchips</i>	Ch.1
9/5		Computer Software: Apps & Operating Sys	Ch.2
9/8		Central Processing Unit Article Review #1 DUE (Popular Literature)	Ch.3
9/12		Input and Output: The User Connection	Ch.4
9/15		Storage and Multimedia Article Review #2 DUE (Major Area of Study)	Ch.5
9/19		Networking: Computer Connections	Ch.6
9/22	X	The Internet: A Resource for All of Us <i>Gallery 2: The Visual Internet</i> Article Review #3 DUE (Technical literature)	Ch.7
9/26	X	The Internet in Business	Ch.8
9/29	X	Writing Your Own Web Page World Wide Web Research Project DUE	Ch.9
10/3	X	Presentation Software: PowerPoint	
10/6	X	Presentation Software: PowerPoint	
10/10		Fall break!!!!	
10/13	X	MIDTERM EXAMINATION	
10/17		Multimedia (PowerPoint) presentations Multimedia Projects DUE	
10/20		Multimedia (PowerPoint) presentations	
10/24		Electronic Spreadsheets: Excel	
10/27		Spreadsheets: Business Graphics: Excel <i>Gallery 3: Computer Graphics</i>	
10/31	X	Database Applications: Access	
11/3	X	Working with Access	

Spreadsheet Project DUE

11/7	Working with Access	
11/10	Programming Languages	Ch.14
11/14	Systems Analysis and Design <i>Gallery 5: Buyer's Guide</i>	Ch.15
11/17	Management Information Systems <i>Gallery 4: Computers at Work</i>	Ch.16
11/21	Ethics and Intellectual Property	
11/24	Thanksgiving Break!!!!	
11/28	Video: The Computer, The KGB, and Me	
12/1	History of Computing	Appendix
12/5	Video: Code Rush: A Year in the Life of a Silicon Valley Super Nova	
12/8	Cutting Edge: Expert Systems and Robotics	Ch.17
12/12-13	Study Days (hint!!!!)	
12/15	Video: Bill Gates and the Future Database Project DUE	

12. An Introductory Informatics Course

Mehmet M. Dalkilic
Indiana University in Bloomington

Abstract

This paper describes the use of historical elements in the introductory class for a new curriculum of Informatics at Indiana University, Bloomington. Some preliminary background is provided to set the context of the class, followed by several examples used in I101. A brief reflection of the benefits of history as a teaching tool is made at the conclusion. Appendix A provides a collection of elements from the class.

In this paper I present some uses of history in I101 Introduction to Informatics and conclude with some reflections on history as a teaching tool. The reader is strongly encouraged to read William Aspray's excellent paper, "Using History in a Social Informatics Curriculum" in chapter 5. His paper not only gives a fascinating account of the use of history in a social informatics curriculum, but also presents a discussion about Informatics itself on both a national level and at Indiana University.

I will first provide some background about I101. In 2000, the Indiana University School of Informatics offered its first classes. Chief among these was its introductory class I101, which obviously would provide a conduit for majors. Just as importantly, I101 would establish what Informatics was as a major, as a vocation, and as a field of research. Several faculty were recruited to teach I101 and were given complete academic freedom. Although there was no sense of competition, Darwinistic principles did seem to come into play. After four years of teaching I101, my content survived—actually flourished—and I was made responsible for the I101 curriculum.

When I was given my first section of I101, I had just completed my Ph.D. in computer science. My area of research was, fortuitously, the use of entropy¹²⁴ in data mining. Being a newly minted Ph.D., I was eager to create my own curriculum and thus set out on my own to spend the summer figuring out what Informatics was. As an avid reader, I decided to look for books—hopefully scholarly ones—that could provide some background, or at least context, for informatics. My initial searches involved looking for any work that contained the word "informatics." To my surprise, I found few books that looked worthwhile, and so I broadened my search to include "information." I began by reading several promising books, some of the most fruitful being:

- *The Information Mosaic* by Sharon McKinnon and William Bruns, Jr.
- *How We Became Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics* by N. Katherine Hayles
- *Information Ecology* by Thomas H. Davenport with Laurence Prusak
- *Infosense* by Keith Devlin
- *The Information Paradox* by John Thorp
- *The Living Company* by Arie de Geus

¹²⁴ Entropy (information-theoretic) is a means of modeling information, most well-known through Claude Shannon's work, by considering how effectively one can encode data that is transmitted between a sender and a receiver. From another perspective, entropy gives an account of how uniform a probability distribution is—the more uniform, the higher the entropy or so-called "surprise."

- *The Thinker's Toolkit: 14 Powerful Techniques for Problem Solving* by Morgan D. Jones
- *Holding on to Reality* by Albert Borgman
- *Universal History of Numbers* by George Ifrah
- *The Situation in Logic* by Jon Barwise

In *The Information Mosaic*, McKinnon and Bruns discover that it is the flow and content of information that truly matters in the running of a successful business. From *The Living Company*, de Geus presents information from a private study conducted by Royal Dutch/Shell¹²⁵ that examined the lifespan of more than two dozen companies. What the author cites is a fascinating validation of the previous authors' statements—that the average life expectancy of a company of four or five decades had to do with how the company managed and successfully used its knowledge, and little else. From the *Information Paradox*, Thorp plays on Solow's observation of IT¹²⁶ by pointing out that it is not the technology, but both the presentation of IT and how it is being used. *The Thinker's Toolkit* is an interesting book that was written by a retired CIA analyst who presents about a dozen techniques that aid in analytical thinking in situations where the problems are generally non-quantitative and ill-defined. *The Universal History of Numbers* is a tome containing the history and use of all known counting systems from earliest times to the present.

What I eventually realized was that Informatics is about how people and technology mix. There are various perspectives¹²⁷ one can take. For example, a *social* perspective studies how technology is impacting human society. I am interested in a *technical* approach, for lack of a better term—how we can successfully apply technology to problems: *problem solving with technology*.

Currently I101 is four credit hours and consists of a biweekly lecture lasting an hour and a quarter and a weekly one-hour laboratory. The lecture and laboratory are loosely coupled, with the lecture focusing on the more difficult aspects of the class (problem solving) while the laboratory concentrates on learning and applying technologies like email, word processing, Unix commands, XML, and web publishing. A typical laboratory has twenty to thirty students and two graduate teaching assistants. In the past three years, we have had, on average, upwards of nearly 150 students in the fall semester and around 60 in the spring.

I believe it is not the technology that is difficult, but understanding how we can successfully apply technology, if it is warranted, that is challenging. To actually do this, students must be intimately acquainted with problem solving itself: they must be able to identify a problem, choose different strategies, be aware of constraints, implement solutions, and measure successfulness. Students learn about the various aspects of problem solving by reworking problems from the past. Throughout the several years that I have taught the class I have found that virtually none of the students had seen any of these problems before—so, from their perspectives, the problems and solutions were novel. I will present several uses of history

¹²⁵ Royal Dutch/Shell, *Planning Corporate Change: A Look at How Long-Established Companies Change* (September 1983).

¹²⁶ The paradoxical ubiquitous presence of IT, but the relative absence of demonstrable positive impacts—the so-called “productivity paradox.”

¹²⁷ There are many exciting perspectives. One has to do with our ability to generate, collect, and cheaply store information, coupled with our belief that if a little information is good, then a lot of information is better. Data mining studies how to extract useful, nontrivial, and latent information from data sources that cannot be examined piecemeal. Another perspective is Human Computer Interaction, which studies how to make the interface between people and technology as effective and efficient as possible.

as it appears in I101. I will first describe the concepts I am aiming for. I'll then present the problem—this is what the student sees.

1. Cro-Magnon IT

Concept: I want the students to start reflecting on what *symbolic* means. One of the most difficult tasks faced by Informatics, or any IT-related field actually, is data representation—how to encode information from the real world and manipulate the encoding digitally to something useful and meaningful. Encoding in this case means some established syntax (structure), semantics (meaning), together with the important benefits, such as verification.

Problem: A notched bone from 19K-21K BCE was found in Landes, France, and a rendering by your professor is shown below. It is one of the first IT devices. What was its function? How might it be improved? What problem is it solving? We're not looking for easy answers like jewelry. A couple of paragraphs would be a good start.

2. A is for Apple

Concept: This is a problem that is modified after a problem in Polya's *How to Solve It* (1957). I'm interested in the students being aware of constraints, both real and imagined. One of the most difficult aspects of this problem is the lack of boundaries, too—the problem is intentionally left vague.

Problem: Look at the following list of letters:

- I. A, M, T, U
- II. B, C, D
- III. N, S, Z
- IV. H, I
- V. F, G, J

Finish adding the capitals (upper-case), then all the lower-case letters, and then digits 0-9. Each symbol can belong to one and only one line. You need to give reasonable explanations why you've placed the symbols as you have. What assumptions have you made? Your success is how convincing your solution is to a random classmate.

3. Narcissus

Concept: This is a problem modified after a Chemtech 22(1) (1992). I'm interested in the students realizing that solutions are not necessarily technological. This famous problem illustrates that not all solutions are obvious either.

Problem: You've just graduated with your Informatics degree and have been hired to help solve a problem. A hotel has been doing such a good business that it has decided to add another 20 floors to its existing 20. Complaints from hotel guests have begun occurring about the time they wait for the elevators. What is the perceived problem? What is the actual problem? Come up with solutions for both the perceived and actual problem and discuss how they differ. How would your solution's success be measured?

Comments: The solution that was eventually employed was simply placing mirrors next to the elevators—the guests spent time looking in the mirrors and consequently lost track of time. The complaints ceased.

4. Mirrors

Concept: This is a problem modified after an article from the *Washington Post* (12/14/92, p A3). By the time students see this problem, they've seen the problem above. The solution is akin to the solution from above.

Problem: This is a real problem that has occurred. You're consulting for an airline that has done a remarkable job in getting passengers deplaned and to the baggage claim. On average, passengers are able to get to baggage claim once deplaning has begun in about five minutes. Bags take about another five to ten minutes. Passengers have begun complaining about waiting times. What is the perceived problem? What is the actual problem? Provide solutions for the actual problem. What are some possible ramifications of your solution? How are you measuring the success of your solution?

5. Epidemiology

Concept: This is a problem fashioned after John Snow's discovery in the mid-1800s of the source of cholera in London, adapted from *Investigating Disease Patterns* by Paul D. Stolley and Tamar Lasky (1995). Although the answer becomes somewhat obvious, it is the power of data representation and visualization that I'm interested in the students becoming sensitive to.

Problem: Cholera is a deadly disease that has ravaged many parts of the world. In 1831 the first case of cholera occurred in the United Kingdom in the city of Sutherland. After symptoms of vomiting, diarrhea, and fever, the patient died. Soon after, cholera began to ravage London. John Snow, one of the earliest Informaticians, sought to figure out how the disease was being transmitted. Snow mapped out various data of the city. Below you'll find data that emulates the data Snow had. On the map provided, mark the occurrences of cholera and other data elements that Snow studied. What conclusion did he come to? (For the sake of space, I've not included the data in this paper).

Conclusion

I've provided only a few cases of the use of history. There are many more—for example, the social impacts of perceived problems as evidenced by A.D.J. Macfarlane's *Witchcraft in Tudor and Stuart England*.

What does history as a tool do for me as an instructor? History provides a context. Especially in a new discipline like Informatics, we can begin to understand what it is by understanding how it relates to other areas—what are the stories? History also provides a foundation—not just a mathematical one, but a place to call home. Context and foundation establish legitimacy. This is particularly important to Informatics and the intellectual experience of the students taking my class. History provides ready-made problems. I believe every course must be new in some important way each time it is taught, but in an area that thrives on change—informatics—this is especially so. Coming up with new material is a daunting, time-consuming task. By using history I have well over several millennia to choose from. History also

provides intellectual roadmaps. I can retrace events in the classroom, knowing full well the outcomes and the pitfalls.

I believe, ultimately, that teaching is a journey and there is no better guide than history. I am including a collection of topics from I101, in no particular order, as Appendix A.

Mehmet M. Dalkilic is an Assistant Professor in the School of Informatics, Adjunct Assistant Professor in Computer Science, and Group Leader at the Center for Genomics and Bioinformatics. He is the curriculum coordinator for both the undergraduate introductory class I101 Introduction to Informatics and graduate bioinformatics introduction class L519. His current research activity is in bioinformatics, data mining, and teaching. *Author's address:* School of Informatics, Indiana University, Bloomington, IN 47408 USA. Email: dalkilic@indiana.edu.

Appendix A: Introduction to Informatics I101-General Syllabus Outline

© 2003 M.M. Dalkilic

Introduction to Informatics I101 is an exciting class that enables the student to learn about and utilize problem solving techniques and technology. There is a computer laboratory that supplements the lecture materials.

Informatics is about problem solving with technology. We investigate what “problem solving” means as a process. We learn how to become better problem solvers by learning and applying different techniques. Students will also learn about IT from the bit to the internet. The laboratory works in concert with the lecture. The laboratory culminates with the creation of a web-based portfolio from which all future work can be posted. The lecture outline and laboratory outline are presented next

Outline of Lectures

Introduction to Informatics

- motivation
- definition
 - o solving problems with technology
 - o adding “structure”
 - * verify
 - * reproduce
 - * atoms + rules for building
 - * syntax-properly built
 - * semantics-what the structure means
- compare & contrast with other disciplines
- challenge is to capture reality digitally-many times faced with capturing analog information with digital
- Analog vs. Digital

Problem Solving

- motivation
- definition
 - o change in state of affairs from current to possible
 - o solution is the path
- problem solving sins
- process of problem solving
- application

Problem Solving-the “Problem Statement”

- motivation
- definition
 - o the “problem statement” delineates the set of solutions pursued from those that aren’t
 - o significance
- techniques to create or discover the correct problem statement
- application

Problem Solving-Algorithms vs. Heuristics

- motivation

- definition
 - desirability of algorithm, likelihood of heuristics
- resources used in problem solving
 - money, people, space, time, ...
- “Brute force” as a first example of algorithm
- “Russian Peasant” multiplication as a second example of algorithm
- Introduction of flow chart elements—another example of structured reasoning
 - $var \leftarrow expression$; **PRINT** var ; **GET** var ;
 - operation; branch; I/O

Capturing Reality Digitally

- motivation
 - images, sound
- definition
- (digital) numbers as a working example
- examine what is a number
 - tally
 - different bases
- convert from base x to base 10—algorithm
- convert from base 10 to base y —algorithm
- draw attention to binary, octal, hex
- can “build” larger algorithm from two smaller algorithms
 - to convert from base x to base y use two algorithms learned previously
- using flow chart show how to convert from base 10 to base x

Problem Solving—Making Language Precise: Propositional Logic (P L)

- motivation
 - natural language vs. artificial (mathematical)
- definition
 - structure + reasoning = logic
 - syntax and semantics
- operators $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- meaning
- truth tables
 - exhaustive listing of input values with output
 - related to digital (count in binary for all possible inputs)
- tautology, contradiction
- translation from English to PL and PL to English

Technology—HTML

- motivation
- definition
- XHTML
 - Syntax and semantics
 - Browsers
 - File vs. image
- TAGS
- File formats

Problem Solving: Algorithm Weighted Ranking

- motivation

- definition
 - means of ordering best solutions by uniformly applying most important criteria
- application

Problem Solving & Technology-Gates

- motivation
- definition
- a different formalization of PL
 - syntax and semantics
- Translate from PL to gates and gates to PL
- Translate from English to gates and gates to English
- means of capturing reality digitally
- example of problems
 - majority vote of three people
 - bit adders
- Introduction to computer architecture
 - CPU
 - Bus
 - Memory (captured with gates)
 - Video

Utility Analysis

- motivation
- definition
- Sets
 - syntax and semantics
 - operators: \cap , \cup , \setminus , powerset
 - predicates: \forall
- Models
 - Deterministic
 - Random
- Probability
 - Sample space
 - Events (subsets)
 - Measure on subsets
 - Build sets using operators
 - Actually PL
 - Every set operator has equivalent PL operator
- Utility function
- Bringing the parts together for Utility analysis

Problem Solving—More Models

- Motivation
- Definition
- “Database”
 - purpose
 - relational model
 - structure
 - queries
- Linear Model (least squares)

- Shannon's Model of Communication
 - o Concept of "interestingness" (no mathematics)
 - o Sender, Encoder, Channel, Decoder, Receiver
 - o Bit as a unit of currency (tie into previous work)
 - o Entropy
 - o Speed vs. Correctness

Technology

- motivation
- Computer
 - o Architecture
- OS
- Networks
 - o motivation
 - o definition
 - o physical vs. logical
 - o Model: OSI
 - o architecture
- Internet
 - o General architecture
 - o

Bringing It Altogether

- Problem solving with technology
 - o Bioinformatics
 - o HCI
 - o New Media
 - o Health Informatics
- New Frontiers
 - o RF tags
 - o "smart paper"
- Social Impacts of IT
 - o Ethics
 - o Law

Outline of Laboratory Sessions

Getting Connected at IU: Mail and MS Outlook

Getting Connected at IU: Accounts

- SSH, SFTP
- Security
- OS'
 - o UNIX (Solaris)
 - o Windows XP
- CFS

Resources: Word Processing

- Notepad, MS Word
- format
- Creating, Editing, Saving, Printing

- Page numbers, Images, Tables

Technology: Steel Account (Solaris)

- UNIX
- Picking a shell
- Commands
- Word processing with Emacs
 - .emacs
- directories
- files

Technology: Web pages on Steel

- HTML
- Creation
- chmod
- mypage

Technology: Web pages on Windows

- HTML
- Creating
- Uploading to Steel
- Downloading from Steel

Technology: Web pages

- Tables
- Colors
- Fonts
- Dynamic Web pages
 - Javascript
 - Images
- Event driven programming
- Flow charts
 - Algorithms

Technology: Web pages

- Java
- Applets

Resources: Spreadsheets

- Excel
- Utility Analysis

Technology: Web pages

- Frames
- Maps
- Resource management

Technology: Web pages

- Portfolio design
- Portfolio implementation

13. Engineering and Computer Science in Action: A Course on “The Structure of Engineering Revolutions”

Eden Miller Medina and David A. Mindell
Massachusetts Institute of Technology

Abstract

This class provides an integrated approach to engineering practice in the real world. Students research the life cycle of a major engineering project, new technology, or startup company from multiple perspectives: technical, economic, political, and cultural. Research involves interviewing inventors, reading laboratory notebooks, evaluating patents, and looking over the shoulders of engineers as they developed today’s technologies. Developed in 1997 as a joint effort between the MIT Electrical Engineering and Computer Science Department and the Program in Science, Technology and Society, “The Structure of Engineering Revolutions” illustrates how history can successfully enrich the education of advanced computer science and engineering students.

MIT launched the flagship course “The Structure of Engineering Revolutions” in 1997 under the joint impetus of the Electrical Engineering and Computer Science (EECS) Department and the Program in Science, Technology, and Society (STS). The course aimed to bridge the gap between the excellent technical education received by MIT EECS undergraduate and graduate students and the methodologies used by anthropologists and historians in STS to study technological development, practice, and culture.

The course began as the brainchild of Professor David A. Mindell, who designed it as a vehicle for teaching students about engineering and computer science practice in the real world. Although MIT graduates exhibited a high degree of technical proficiency when they entered the job market, faculty and alumni had expressed concern that, in general, EECS students lacked an overall understanding of how non-technical factors contributed to the successful development of new technologies. Students, they felt, needed an appreciation for the diverse roles engineers played and a deep understanding of engineering cultures and their effects on design. Faculty also expressed a desire for students to increase their experience working in teams, improve their skill giving oral presentations, and be able to think critically and argue across disciplines.

Building on these perceived deficiencies in the EECS curriculum, Mindell created a course that allowed students to gain a better understanding of engineering practice by conducting original research on the history of a technology, company, or individual scientist or engineer. Through these historical case studies, students learned about the day-to-day realities of engineering as documented in laboratory notebooks, personal interviews with project participants, archival sources, and the technologies themselves.

The course targeted advanced students, typically fifth year students completing a Masters in Engineering. In theory, the fifth year was designed to “broaden” a student’s educational experience, but in practice students were using the additional year to deepen their expertise in a particular technical area. “The Structure of Engineering Revolutions” addressed this gap between theory and practice by giving students the opportunity to use the skills honed in their other technical classes and apply them within a different analytical framework.

The course fulfilled a technical requirement within the EECS curriculum. It was not an add-on to a technical education, but part of the core, technical and non-technical, of what it means to be educated as an Electrical Engineer or Computer Scientist at MIT. Support from the EECS department for the educational goals of the class contributed significantly to its success and helped with enrollment levels while the class was still a relatively unknown entity. To encourage the interdisciplinary merging of EECS and STS, the course has frequently been taught jointly by instructors from both departments. Mindell is both an historian of technology and a practicing electrical engineer, and every teaching assistant involved with the course has possessed a dual background in history and EECS.

Course Themes

Classical engineering and science education encourages students to search for the best technical solution. The typical problem set presents a situation with a series of controlled conditions and one right answer. Nothing exists outside of this technical realm, or if it does it is viewed as unimportant, suspicious, or even bad science or engineering. This emphasis on the technical fosters a popular myth among students that simply building the next killer app or disruptive technology will automatically lead to fame and riches. However, as history has shown time and time again, the successful development, implementation, and marketing of new technologies depend on a broad range of skills, many of which are often dismissed as politics.

The material presented in “The Structure of Engineering Revolutions” encourages students to break down the artificial barrier between the “technical” and the “political” and learn that politics exists in every aspect of technological production, from the creation of an engineering culture to determining the criteria that define reliability and success. These political, or perhaps social, considerations are not irrational as students may initially believe, but rather reflect the rationality of a particular engineering culture, market, or set of social relationships.

If we take this as a starting point, it then becomes easy to see how history and the interdisciplinary field of Science, Technology and Society can provide a crucial addition to engineering and computer science education.

To cite a few examples:

- Instead of thinking about technology and its social effects, a deterministic model that places human beings in the position of helplessly conforming to the introduction of new technologies, students learn that the technologies themselves are social creations, brought about by a particular historical context, series of negotiations, and individual choices.
- Students learn to question technological trajectories such as Moore’s Law and analyze the complex socio-technical networks that permit these trajectories to masquerade as the inevitable path of progress.
- We study a variety of engineering cultures (e.g., Microsoft, Data General, and the MIT Instrumentation Lab), and through these case examples illustrate how managers historically have cultivated particular working conditions in order to shape engineering practices, technological design, and company identity. By studying the heterogeneous factors that contribute to and maintain a particular engineering culture, students learn to think beyond the strictly technical sphere of the lab and ask instead how their work environment has been constructed and the effect it has on the work produced.

- We take apart the myth of the single genius inventor, the origin story, and idea of “vision.” History clearly shows that invention and innovation are processes rather than individual moments. Inventors depend upon laboratory, government, or other social or institutional infrastructures to create their famous works, rather than simply relying on genius. Likewise, visionaries do not convey prophecies of what will be. They convince others to share their vision and help make their interpretation of the future into a reality.
- In contrast to the problem set model that places technical considerations above all, students learn to put people, their choices, and their decisions at the heart of technological development.

Course Structure

The course is divided into two parts. The first half follows a traditional format with lectures, readings, and short writing assignments akin to problem sets that introduce students to the themes mentioned above and permit early feedback on their writing skills.

The syllabus also includes a workshop on working in teams, a lecture from a patent lawyer on the basics of reading a patent, and a presentation by the MIT archives on how to access and incorporate the rich collection of laboratory notebooks and other primary source documentation in students’ final projects. A unit based on Edward Tufte’s work on visual representation encourages students to think critically about the way they use and display visual information in their written work and oral presentations.

In the second half of the course, the students divide up into teams and write their own case histories of a technology, company, or individual of their choosing. This is similar to the case method used in business schools, except the students have the opportunity to write the cases themselves and select the research questions they find most interesting. In addition to presenting a well-reasoned argument, students are required to demonstrate a solid technical understanding of the technologies relevant to their case history. Students typically worked in teams of four to six students and presented their work in a public forum on the MIT campus at the end of the semester.

The second half of the course is largely unstructured, but nonetheless time-intensive for both students and teachers. Although students were free to use class time to visit the library, conduct interviews, or have group meetings, we maintained a hands-on relationship with each group, helping them frame research questions and suggesting additional source materials. As the projects advanced, we provided detailed critiques of written drafts and oral presentations. Due to the amount of time invested in each student, the interactive style of class lectures, the need to limit groups to six students (four is preferable), and the requirement that each group publicly present their work in an open forum at the end of the semester, we have not been able to scale the course beyond fifty students. An enrollment of sixteen to twenty students is ideal.

From the student perspective, the work assigned for this class rivaled the workloads found in their other EECS classes, especially during the final phase of the semester. For this reason, in later syllabus revisions we decided to give students more class time to work on their final projects. The projects also introduced students to the numerous benefits and difficulties of working in teams. Assignment guidelines required students to produce a seamless final document that did not illustrate an obvious division in labor and/or writing style. Some groups also opted to construct a project website in addition to the final written document.

Despite the historical framing used in the course, the goal was not to teach students to become historians of technology; nor was it to impart any particular historical narrative. Rather, history served as a vehicle to teach an awareness of complexity in technology and to impart critical reading and thinking skills. The projects that students examined contained all the contradiction, ambiguity, and uncertainty of historical accounts. In resolving these difficulties, students learned to identify the broad parameters of engineering practice and developed a conceptual vocabulary with which to describe the evolution of a technology to themselves and to others.

By the end of the semester, the students had created a sophisticated historical analysis of engineering and computer science in action that drew upon a broad range of primary and secondary source materials. While many educators fear the increasing and uncritical dependence on Google exhibited by students in recent years, the course structure provided an opportunity for students to utilize a combination of interviews, standards, patents, laboratory notebooks, history texts, technical diagrams, code, technological artifacts, and photographs in the creation of an original historical narrative and argument.

Apart from these educational objectives, the course augmented ties among students, alumni, local businesses, and various university resources. By requiring students to select a project history associated with MIT in some form, we encouraged interaction among the students and these diverse university and technological communities with positive results. University alumni have enjoyed contributing to student education outside of annual giving campaigns. Project interviews can similarly lead to job interviews for students or create networks for the future. The diverse sources used in writing the project histories also allowed under-funded but important resources like university archives to claim a visible contribution to undergraduate and graduate teaching. These positive aspects of community-building could potentially be transferred to other universities by implementing a modified version of the course curriculum.

Conclusion

Young engineers and computer scientists face a world that demands a multidisciplinary approach to engineering and computer science, where technology does not develop in a vacuum. It is a world in which graduates will spend a majority of their time communicating and where they must exercise judgments based on diverse and imprecise information. The themes addressed in this class encourage students to synthesize their strong technical skills with insight into political and social processes, abilities in teamwork, communications, and career-long learning. This last is perhaps the most subtle and difficult to teach; it must combine the thrill of learning with the ability to read, think, and write critically across disciplines. Here the disciplines and methods of history and STS have the most to contribute.

MIT and the Boston area have provided an environment rich in resources for the themes and ideas presented in this class, but they are not unique. Every engineering and computer science department boasts a history of technological innovation, an alumni network, and a local academic or industrial high-tech community. Since this class encourages students to situate engineering and computer science practices in a real world environment, departments may choose to couple this class with a pre-existing professional practice requirement or internship program, perhaps encouraging students to select projects of relevance to their future employers. Restructured, this course could address histories and practices in science and engineering fields outside of EECS. It could also be tailored to stress the particular research and teaching strengths of a given university or EECS department, while still encouraging students to

develop and apply a range of interdisciplinary skills outside of those found in the existing curriculum. For universities hoping to enrich their humanities curriculum with facets of science and engineering education, the human-centered approach to technology building presented here could appeal to humanities students and allow them to think about technology-related issues at a high level of sophistication.

“The Structure of Engineering Revolutions” has consistently received positive student reviews, with some students claiming it has been the most valuable course of their MIT experience, and has resulted in the university’s highest teaching award for Mindell. Interested readers are encouraged to visit the course website (<http://mit.edu/6.933/www>) for links to class handouts, notes, and copies of final project histories. Material from “The Structure of Engineering Revolutions” is also available publicly as part of MIT OpenCourseWare (<http://ocw.mit.edu/index.html>).

Eden Miller Medina is a Ph.D. Candidate in the MIT Program for Science, Technology and Society and an electrical engineer. She has been involved with “The Structure of Engineering Revolutions” as both teaching assistant and student for three of the five years it has been offered. She is currently finishing a dissertation on the history of computer development in Chile during the 1960s and 1970s and will begin her appointment as an Assistant Professor in the School of Informatics at Indiana University in 2005. *Author’s address:* Program in Science Technology and Society, MIT E51-070, 77 Mass Ave, Cambridge, MA 02139 USA. Email: eden@mit.edu.

David A. Mindell is the Frances and David Dibner Associate Professor of the History of Engineering and Manufacturing in the Program for Science, Technology and Society at MIT. His research interests include technology policy (historical and current), the history of automation in the military, the history of electronics and computing, and deep-sea archaeology. Professor Mindell heads MIT’s “DeepArch” research group in Deep Sea archaeology. He is the author of *War, Technology and Experience Aboard the USS Monitor* (2000), and *Between Human and Machine: Feedback, Control, and Computing before Cybernetics* (2002). *Author’s address:* Program in Science Technology and Society, MIT E51-194C, 77 Mass Ave, Cambridge, MA 02139 USA. Email: mindell@mit.edu.

Appendix A. List of Final Projects Relevant to Computer Science¹²⁸

Fall 2001

- Mathematical Theory of Claude Shannon: A Study of the Style and Context of his Work Up to the Genesis of Information Theory
- Information Theory and the Digital Age
- The MIT Student Information Processing Board: The Social and Technical Impact of an MIT Student Group
- Bolt Beranek and Newman Inc.: A Case History of Transition
- A Marriage of Convenience: The Founding of the MIT Artificial Intelligence Laboratory

Fall 2000

- Down from the Top of its Game: The Story of Infocom, Inc.
- The World Wide Web as an Engineering Paradigm
- The Selection of the Advanced Encryption Standard
- Mode S: An Air Traffic Control Data-Link Technology
- LEGO Mindstorms: The Structure of an Engineering (R)evolution

Fall 1999

- Project Athena: Success in an Engineering Project
- Logo: A Project History

Fall 1998

- Ethernet: An Engineering Paradigm
- Symbolics, Inc.: A Failure of Heterogeneous Engineering
- Dragon Systems

Fall 1997

- The Apollo Guidance Computer
- Thinking Machines
- Lotus
- Magnetic Core Memory
- RSA Data Encryption

¹²⁸A full list of projects is available on the course website, http://mit.edu/6.933/www/projects_whole.html.

Appendix B. Course Syllabus

6.933J / STS.420J The Structure of Engineering Revolutions Course Outline Fall, 2001

I. Introduction and Background

September 5: (Wed)
Introduction, course overview.

II. Engineers in Action

September 10: (Mon)
Reading: Latour, Science in Action Introduction (pp.1-17), Chapter 3 (pp. 103-144)
September 12: (Wed)
Reading: Latour, Science in Action Chapters 4-6 (pp. 145-259)

III. The Construction of Technological Systems

September 17: (Mon) NO CLASS
September 19: (Wed)
Reading: Mindell, "Opening Black's box: Rethinking feedback's myth of origin" (Xerox/pdf)

IV. Invention and Engineering Culture

September 24: (Mon)
Reading: MacKenzie, Inventing Accuracy Chapters 1-2
September 26: (Wed)
Reading: MacKenzie, Inventing Accuracy Chapters 4, 7, 8.
Assignment Due: Response paper #1

V. Innovation and Marketplace

October 1 (Mon)
Reading: Christensen, The Inventor's Dilemma Chapters 1, 2, 4, 9, 11

October 3: (Mon)
Reading: Discussion of methodology, research methods, library, techniques, source materials, etc.
Assignment Due: Response Paper #2

VI. Using Sources Effectively

October 8: (Mon) NO CLASS - COLUMBUS DAY

October 10: (Wed) **Lab Notebooks and Project Documentation**
Meet in MIT Archives 14N-118, introduction to Edgerton/Forrester notebooks

VII. Group Work and Collaborative Writing

October 15: (Mon) Presentation "Working effectively in groups", in class group work
October 17: (Wed) In Class Group Work, Presentation of project history proposals, discussion and ranking.
Assignment Due: Individual project history proposals

VIII. Project History Work

October, 22: (Mon) In class group work
October, 25: (Wed) In class group work

IX. Visual materials and argumentation

October 29: (Mon)
Reading: Edward Tufte, Visual and Statistical Thinking (packet)
October 31: (Wed) In class group work

X. Understanding Patents

November 5: (Mon) Presentation "How to read a patent", in class group work
November 7: (Wed) In class group work
Assignment Due: Visual argumentation exercise

XI. Project Histories

November 12: (Mon) NO CLASS
November 14: (Wed) In class group work
Assignment Due: Group proposal/plan of project history
November 19: (Mon) In class group work
November 21: (Wed) In class group work

THANKSGIVING

November 26: (Mon) In class group work
November 28: (Wed) In class group work

XII. Final Preparations

December 3: (Mon) Presentation rehearsals

December 5: (Wed) Presentation rehearsals

Assignment Due: Written draft for groups presenting

XIII. Presentations of Project Histories

December 10: (Mon) Group Presentation of Project Histories

December 12: (Wed) Group Presentation of Project Histories

LAST DAY OF CLASSES

Final projects due Friday December 14, 5pm.

Additional Course Syllabi (chapters 14-20)

14. STS.035: From Analog to Agents: Introduction to the History of Computing (Course Syllabus)

David A. Mindell
Massachusetts Institute of Technology

David A. Mindell is the Frances and David Dibner Associate Professor of the History of Engineering and Manufacturing in the Program for Science, Technology and Society at MIT. His research interests include technology policy (historical and current), the history of automation in the military, the history of electronics and computing, and deep-sea archaeology. Professor Mindell heads MIT's "DeepArch" research group in Deep Sea archaeology. He is the author of *War, Technology and Experience Aboard the USS Monitor* (2000), and *Between Human and Machine: Feedback, Control, and Computing before Cybernetics* (2002). *Author's address:* Program in Science Technology and Society, MIT E51-194C, 77 Mass Ave, Cambridge, MA 02139 USA. Email: mindell@mit.edu.

STS.035
From Analog to Agents:
Introduction to the History of Computing
Massachusetts Institute of Technology
Spring 1998

Professor David A. Mindell

Program in Science, Technology, and Society

Building E51 Room 194A

Phone: x3-0221, mindell@mit.edu

Office Hours: Mondays 2-4 pm (or make appointment through Judy Spitzer, x3-4044)

Course web site: *web.mit.edu/STS.035/www/*

Class meets: *Monday & Wednesday, 11-12:30 in room 1-135. Attendance at lectures is required.*

Course Description: *Examines the development of computing techniques and technology in the nineteenth and twentieth centuries, particularly critical evaluation of how the very idea of “computer” changes and evolves over time. Emphasis is on technical innovation, industrial development, social context, and the role of government. Topics include Babbage, Hollerith, differential analyzers, control systems, ENIAC, radar, operations research, computers as scientific instruments, the rise of “computer science,” artificial intelligence, personal computers, and networks. Includes class visits by members of the MIT community who have made important historical contributions.*

Prerequisites: *General familiarity with computer architecture, some prior programming experience.*

Grading:

3 quizzes @ 20 % each 60%

Final paper 30%

Attendance and participation 10%

Quizzes: *There will be three quizzes over the course of the term, lasting about half of a class period. Quizzes will most likely be a series of identifications of terms from lectures and readings and are intended to be straightforward if you are keeping up the work in the class. March 2, April 15, May 11.*

Final Paper: *An 8-10 page paper is due on the last day of class. More will be said about it when the time comes, but it should be a detailed discussion of some issue or topic raised during the course. Simulation projects of historical computers or web projects are also potentially acceptable, pending approval of the instructor. A one-page proposal/outline of the final paper will be due in mid-April, to allow for instructor feedback. Writing assignments will be graded on force of argument, clarity of presentation and relevance to course material. Proper citation practices should be followed throughout (ask if you are unsure of the details).*

Course Books (on sale in the Coop):

Martin Campbell-Kelley and William Aspray, *Computer: A History of the Information Machine* (NY: Basic, 1996).

Claude Shannon and Warren Weaver, *The Mathematical Theory of Communication* (Urbana: U. of Illinois Press, 1949, repr. 1963).

Frederick Brooks, *The Mythical Man Month: Essays on Software Engineering -- 20th Anniversary Ed.* (Reading, MA: Addison-Wesley, 1995).

Kurt Vonnegut, *Player Piano* (NY: Dell, 1988).

NOTE: Other required readings are available as a packet on sale at MIT CopyTech Center (11-004); some are downloadable from web sites as indicated. The course books, as well as the optional readings, will be placed on "library reserve."

Weekly Syllabus: (*=readings in packet)

February 4: Introduction and overview, early history

*Mahoney, "The History of Computing and the History of Technology."
Computer (Chapter 1).

February 9: Industry: Babbage's philosophy of computing

*Charles Babbage, "Difference Engine No. 1" and "Of the Analytical Engine." Chapters V & VIII from *Passages from the Life of a Philosopher* ;

**"On the Division of Mental Labour." Chapter XIX from *Economy of Manufactures and Machinery*.

Also see: <http://socserv2.socsci.mcmaster.ca:80/~econ/ugcm/3ll3/babbage/index.html>

*Simon Schaffer, "Babbage's Intelligence: Calculating Engines and the Factory System."

February 11: Industry: Hollerith and mechanical "information processing"

*James Beniger, *The Control Revolution: Technological and Economic Origins of the Information Society -- Chapter*.

Computer (Chapter 2).

February 17 (Tuesday): Analog Computing: Fire Control & Auto Pilots

*Mindell, "Anti-Aircraft Fire Control and the Development of Integrated Systems at Sperry, 1925-1940."

February 18: Analog Computing: The Differential Analyzer

*Vannevar Bush, "The Differential Analyzer: A New Machine for Solving Differential Equations."

*Larry Owens, "Vannevar Bush and the Differential Analyzer: The Text and Context of an Early Computer."

February 23: Military: World War II and Military Applications

Computer (Chapter 4).

*Mindell, "Engineers, Psychologists, and Administrators: Control Systems Research in Wartime, 1940-1945."

**"Electrical Computers for Fire Control."

February 25: Systems: The Shift to Digital

*Larry Owens, "Where are We Going, Phil Morse?: Changing Agendas and the Rhetoric of Obviousness in the Transformation of Computing at MIT, 1939-1957."

*Claude E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits."

March 2: QUIZ #1

Military: Cryptography and Colossus

*Samuel Snyder, "Computer Advances Pioneered by Cryptologic Organizations."

March 4: Systems: Toward the Stored Program: ENIAC and EDVAC

*A.M. Turing, "On Computable Numbers" [1937].

*John von Neumann, "First Draft of a Report on the EDVAC" [1945].

*Paul Ceruzzi. "Crossing the Divide: Architectural Issues and the Emergence of the Stored Program Computer."

Web page on ENIAC history: <http://www.seas.upenn.edu/~museum/hist-overview.html> ; also <http://ftp.arl.mil/~mike/comphist/>

March 9: Human/Machine: Feedback, Control, and Computing

*Mindell, "Feedback Amplifiers and Mixed Emotions at Bell Telephone Laboratories" and "Conclusion."

March 11: Communications: The Theory of Information

*Claude Shannon and Warren Weaver, *The Mathematical Theory of Communication*

*William Aspray, "The Scientific Conceptualization of Information: A Survey."

March 16: Computer Science: Norbert Wiener and Cybernetics

*Rosenblueth, Bigelow, Wiener, "Behavior, Purpose, and Teleology."

*Peter Galison, "The Ontology of the Enemy: Norbert Wiener and the Cybernetic Vision."

Optional:

Norbert Wiener, *Cybernetics; The Human Use of Human Beings*; God & Golem Inc..

Donna Haraway, "A Cyborg Manifesto: Science, Technology, and Socialist-Feminism in the Late Twentieth Century," in *Simians, Cyborgs, and Women*.

March 18: (video)

[Spring Break]

March 30: Human/Machine: Project Whirlwind and Real Time (Guest, Jay Forrester?)

Computer (Chapter 7).

*Thomas M. Smith, "Project Whirlwind: An Unorthodox Development Project."

April 1: Military: The Cold War and Large Systems

*Kenneth Flamm, "Military Roots," in *Creating the Computer*.

*Robert Everett, Charles Zraket, and Herbert Bennington, "SAGE -- A Data Processing System for Air Defense."

Optional:

Kent Redmond and Thomas Smith. *Project Whirlwind*.

Donald MacKenzie, "The Influence of the Los Alamos and Livermore National Laboratories on Supercomputing."

April 6: Industry: Automation and ‘Social Implications’

Kurt Vonnegut, Player Piano.

Optional:

Shoshanna Zuboff, In the Age of the Smart Machine.

David Noble, “Social Choice in Machine Design: The Case of Automatically Controlled Machine Tools,” in Case Studies on the Labor Process.

April 8: Computer Science: The Rise of Software

Frederick Brooks, The Mythical Man Month (20th Anniversary edition; Addison Wesley).

*Michael Mahoney, “The Roots of Software Engineering”

April 13: Industry: Origins of the Computer Industry

*Emerson Pugh and William Aspray, “Creating the Computer Industry.”

*Duncan Copeland, et. al., “Sabre: The Development of Information-Based Competence and Execution of Information-Based Competition.”

April 15: QUIZ #2

Human/Machine: Real-Time, Control, Aviation

*James Tomayko, "Nasa's Manned Spacecraft Computers."

*Norman Sears, “Technical Development Status of Apollo Guidance and Navigation.”

April 20 (No class, Patriot’s day)

April 22: One-page proposals for final projects/papers due

Systems: Timesharing

*Karl Wildes and Nilo Lindgren, Chapter 22: “The Evolution of Time Sharing.”

Multics History Page: <http://www.best.com/~thvv/history.html>

Optional:

Dennis Ritchie, “The Evolution of the UNIX Time-Sharing System.”

April 27: Human/Machine: Toward Networking

Vannevar Bush, “As We May Think,” Atlantic Monthly, July 1945. Available at course web site.

J.C.R. Licklider, “Man-Computer Symbiosis” Available at course web site.

*Susan Barnes, “Douglas Carl Engelbart: Developing the Underlying Concepts for Contemporary Computing.”

April 29: Computer Science: Languages

*J.W. Backus, et. al., “The FORTRAN Automatic Coding System”

*Peter Naur, et al., “Revised Report on the Algorithmic Language ALGOL 60” [1960]

Optional:

John Backus: “The History of Fortran I, II, and III,”

John McCarthy, “History of LISP,” both in History of Programming Languages

May 4: Industry: Microprocessors & The PC Revolution

Computer (Chapters 9-11).

*Robert Noyce and Marcian Hoff, "A History of Microprocessor Development at Intel."

Optional

Robert Cringely, Accidental Empires: How the Boys of Silicon Valley Make Their Millions, Battle Foreign Competition, and Still Can't Get a Date.

Paul Freiberger and Michael Swaine, Fire in the Valley: The Making of the Personal Computer.

E. Braun and S. MacDonald, Revolution in Miniature: The History and Impact of Semiconductor Electronics.

May 6: Computer Science: Artificial Intelligence

*A.M. Turing, "Computing Machinery and Intelligence" [1949]

*Arthur Norberg and Judy O'Neill, Chapter 5: "The Search for Intelligent Systems."

*Paul Edwards, Chapter 8: "Constructing Artificial Intelligence"

May 11: QUIZ #3

The Rise of Computer Science

*Michael Mahoney, "Computer Science: The Search for a Mathematical Theory."

May 13: Final Projects Due

Communications: Origins of the Internet

*Norberg & O'Neill, Chapter 4: "Improving Connections Among Researchers: The Development of Packet-Switching Computer Networks."

Useful Web Sites:

Paul Pierce's Computer Collection:

<http://www.teleport.com/~prp/collect/index.html>

Aberdeen Ballistics Research Lab Computing History Site:

<http://ftp.arl.mil/~mike/comphist/>

Smithsonian Computer History (good interviews):

<http://www.si.edu/resource/tours/comphist/computer.htm>

Microcomputer Chronology:

<http://www1.islandnet.com/~kpolsson/comphist.htm>

Short biography of Von Neumann:

<http://ei.cs.vt.edu/~history/VonNeumann.html>

General History with lots of links:

http://www.chac.org/chac/chhistpg.html#HistPg_HdweMGenI

Babbage Institute:

<http://www.cbi.umn.edu/index.html>

Optional / On Reserve:

E. Braun and S. MacDonald, Revolution in Miniature: The History and Impact of Semiconductor Electronics (NY: Cambridge U. Press, 1978).

Robert Cringely, Accidental Empires: How the Boys of Silicon Valley Make Their Millions, Battle Foreign Competition, and Still Can't Get a Date (Reading, MA: Addison-Wesley, 1992).

Paul Edwards, The Closed World: Computers and the Politics of Discourse in Cold War America (MIT Press, 1996).

Paul Freiberger and Michael Swaine, Fire in the Valley: The Making of the Personal Computer (Berkeley: Osborne/McGraw-Hill, 1984).

Donna Haraway, Simians, Cyborgs, and Women (NY: Routledge: 1991).

Donald MacKenzie, "The Influence of the Los Alamos and Livermore National Labs. on Supercomputing," *IEEE Annals of the History of Computing* 13, #2 (1991), pp. 179-201.

Kent Redmond and Thomas Smith, *Project Whirlwind* (Bedford, MA: Digital Press, 1980).

Dennis Ritchie, "The Evolution of the UNIX Time-Sharing System," in P. LaPlante, ed., *Great Papers in Computer Science* (IEEE Press / Minneapolis: West Pub. Co., 1996), pp. 705-17.

Martin Van Creveld, *Command in War* (Harvard U. Press, 1985).

M.H. Weik, *A Survey of Domestic Electronic Digital Computing Systems* (Aberdeen, Md.: Army Ballistic Research Lab, 1955; '57; '61).

Richard Wexelblat, ed., *History of Programming Languages* (NY: Academic Press, 1981).

Norbert Wiener, *God & Golem, Inc.* (MIT Press, 1966).

Norbert Wiener, *Cybernetics* (MIT Press, 1965 [1948]).

Norbert Wiener, *The Human Use of Human Beings* (Garden City, NY: Doubleday, 1954).

Andrew Zimbalist, ed., *Case Studies on the Labor Process* (NY: Monthly Review Press, 1979).

Shoshana Zuboff, *In the Age of the Smart Machine: The Future of Work and Power* (NY: Basic Books, 1988).

15. Informatics 303: Organizational Informatics

William Aspray
Indiana University in Bloomington

William Aspray is Rudy Professor of Informatics at Indiana University in Bloomington. He also holds faculty appointments in the computer science department, the history and philosophy of science department, and the School of Library and Information Science. He has served as Associate Director of the Charles Babbage Institute for the History of Information Processing, director of the IEEE Center for the History of Electrical Engineering, and executive director of Computing Research Association. He has also taught at Harvard, Penn, Rutgers, Virginia Tech, and Williams. He serves on the history committees of ACM, IFIP, and SIAM. Current scholarship is focused on policy and historical issues relating to information technology. Author's address: School of Informatics, 901 E. Tenth St., Indiana University, Bloomington, IN 47405 USA. Email: waspray@indiana.edu.

Informatics 303 Organizational Informatics (Spring 2003)

Course Description

This course examines the various needs, uses, and consequences of information in organizational contexts. Topics include organizational types and characteristics, functional areas and business processes, information-based products and services, the use of and redefining role of information technology, the changing character of work life and organizational practices, sociotechnical structures, and the rise and transformation of information-based industries.

Course Organization

The use of information systems to carry out work in various kinds of organizations will be the general organizing theme for the course.

The course has three components:

1. Lectures on various topics in organizational informatics, including:
 - a. Information science analysis of electronic recordkeeping systems
 - b. Business history analysis of the origins and structures of the computer and related industries
 - c. Sociological analysis of the acceptance and use of IT in organizations
 - d. Economic analysis of the management and use of IT in organizations
 - e. Policy analysis of the IT workforce
 - f. Organizational management analysis of the uses of information systems in various industries and in the non-profit sector
2. Guest lectures by practitioners presenting real-world perspectives on the use of IT in organizations
3. Student self-study of the basic principles of business information systems

Required Textbook

The assigned textbook is Steven Alter, *Information Systems: The Foundation of E-Business* 4th ed., Prentice-Hall, 2002. Although the book is expensive, you are strongly encouraged to buy your own copy. Make sure that you buy the 4th edition! We will be reading the entire book.

Quiz and Exam Policy

Material to be tested on quizzes or exams: all material in the assigned readings, plus all material covered in class by the instructor or guest lecturers

Quizzes: We will only count 10 of the 13 quiz scores. Thus if students miss three or fewer quizzes for reasons of illness or religious observance, makeup quizzes will not be given. If a student misses more than three quizzes for reasons of illness or religious observance and can provide written documentation of their reasons, makeups will be given for the number in excess of three that are excused misses. For any student who has taken more than ten quizzes, the ten highest grades will be counted.

Exams: In-class Exams 1 and 2 and the Final Exam.

Why we give so many quizzes and exams:

1. Testing makes you keep up with the material; and students who keep up with the material learn it better than those who cram.
2. Testing reinforces your understanding of the material –you have an active rather than a passive knowledge of the material.
3. The amount of material you need to learn for any one test is reduced –making your study efforts easier.
4. A few poor performances on quizzes (low scores, missed quizzes) can be discounted because we recognize that students do have occasional times when they can not pay full attention to the course (excessively busy with another course, ill, away on job recruitment or family emergencies, etc.);
5. No one quiz or exam counts that much towards the final grade, so a poor performance in one test can be overcome.

Grading and Student Misconduct

The course grade will be determined as follows:

- 25% Exam 1
- 25% Exam 2
- 25% Quizzes (adjusted for participation in discussion section)
- 25% Final Exam

There will be one extra-credit assignment. Details will be discussed in class.

Class schedule:

1. M Jan 13 Course Organization, What Organizational Informatics Is
2. W Jan 15 Computer Industry I (Predecessor Businesses)
3. F Jan 17 discussion –no quiz
4. M Jan 20 no class Martin Luther King Day
5. W Jan 22 Guest Lecturer Philip Bantin (University Archivist, IU) - Electronic Records I (Basic Concepts)
6. F Jan 24 discussion –Quiz: Alter Ch. 1 (E-Business)
7. M Jan 27 Computer Industry II (Mainframe Computers)
8. W Jan 29 Guest Lecturer Jennifer Kurtz (Director eCommerce Office, Indiana Department of Commerce)
9. F Jan 31 discussion –Quiz: Alter Ch. 2 (Business Systems)
10. M Feb 3 Computer Industry III (Mainframes continued)
11. W Feb 5 Guest Lecturer Dennis Morrison (CEO, Center for Behavioral Health)
12. F. Feb 7 discussion –Quiz: Alter Ch. 3 (Business Processes)
13. M Feb 10 Guest Lecturer Philip Bantin –Electronic Records II (Recordkeeping Systems)
14. W Feb 12 Guest Lecturer Julia Sutherland (Director, Technical Services, Monroe County)
15. F Feb 14 discussion –Quiz: Alter Ch. 4 (Databases)
16. M Feb 17 Computer Industry IV (Minicomputers and Microcomputers)
17. W Feb 19 Guest Lecturer Mark Bruhn (Chief IT Security and Policy Officer, IU)
18. F Feb 21 discussion –Quiz: Alter Ch. 5 (Information System s)
19. M Feb 24 Computer Industry V (Software and Services)
20. W Feb 26 EXAM 1 (covers material January 13 through February 17)
21. F Feb 28 discussion –Quiz: Alter Ch. 6 (Customers and Products)

22. M Mar 3 Computer Industry VI (Internet)
23. W Mar 5 Guest Lecturer Alan Minton (Cornerstone Info Systems)
24. F Mar 7 discussion –Quiz: Alter Ch. 7 (Ethics)
25. M Mar 10 IT Workforce
26. W Mar 12 Guest Lecturer Steve Erlich (Aprimo)
27. F Mar 14 discussion –Quiz: Alter Ch. 8 (Networks)
28. M Mar 17 no class –Spring Break
29. W Mar 19 no class –Spring Break
30. F Mar 21 no discussion –Spring Break
31. M Mar 24 Economics and Business I (Productivity Paradox - the Classical Problem)
32. W Mar 26 Guest Lecturer Marlin Eller (SunHawk)
33. F Mar 28 discussion –Quiz: Alter Ch. 9 (Software)
34. M Mar 31 Economics and Business II (Productivity Paradox - Recent Scholarship)
35. W Apr 2 Guest lecturer Herb Senft (1stBooks)
36. F Apr 4 discussion –Quiz: Alter Ch. 10 (Telecommunications)
37. M Apr 7 EXAM 2 (covers material from February 19 through March 31)
38. W Apr 9 Sociology I (Case Study –Alpha Corporation)
39. F Apr 11 discussion –Quiz: Alter Ch. 11 (Information Systems Planning)
40. M Apr 14 Sociology II (Zeta Corporation)
41. W Apr 16 Guest Lecture - Martin Siegel (IU and WisdomTools)
42. F Apr 18 discussion –Quiz: Alter Ch. 12 (Building Information Systems)
43. M Apr 21 Sociology III (Zeta Corp. cont.)
44. W Apr 23 Extra-credit presentations
45. F Apr 25 discussion –Quiz: Alter Ch. 13 (E-Business Security)
46. M Apr 28 TBD
47. W Apr 30 Remaining extra-credit presentations
48. F May 2 discussion –no quiz - Final Exam Review
49. W May 7 (12:30 –2:30) FINAL EXAM

16. Informatics 400/590: Internet and Society

William Aspray
Indiana University in Bloomington

William Aspray is Rudy Professor of Informatics at Indiana University in Bloomington. He also holds faculty appointments in the computer science department, the history and philosophy of science department, and the School of Library and Information Science. He has served as Associate Director of the Charles Babbage Institute for the History of Information Processing, director of the IEEE Center for the History of Electrical Engineering, and executive director of Computing Research Association. He has also taught at Harvard, Penn, Rutgers, Virginia Tech, and Williams. He serves on the history committees of ACM, IFIP, and SIAM. Current scholarship is focused on policy and historical issues relating to information technology. Author's address: School of Informatics, 901 E. Tenth St., Indiana University, Bloomington, IN 47405 USA. Email: waspray@indiana.edu.

Informatics 400/590 Internet and Society (Fall 2004)

Course Description

This course will cover the origins, governance, economics, operation, and use of the Internet. Governance and economic issues include who makes decisions about domain name and spectrum allocation, technical standards and innovation (IPV6, innovation in an end-to-end architectural environment, open and closed standards), and how the Internet is paid for. Operational issues include e spam, worms, viruses, distributed denial of service, accommodation for those individuals with various impairments, and accessibility to a wide range of individuals (Digital Divide issues). Use issues include cybersquatting and trademark infringement, peer-to-peer music- and video-sharing, copyright and copyleft, use of the Internet for vices such as gambling and pornography, e-commerce and taxation, use by the government for e-voting and improved government services, and privacy considerations. Although some technical issues will be discussed, the focus will primarily be on social, political, legal, economic, and historical examination.

Admission requirements: Junior, senior, or graduate school standing, enrollment is limited to 25.

Books to Buy (for both Informatics 400 and Informatics 590 students).

Janet Abbate, Inventing the Internet (MIT Press, 2000) paperback

Milton Mueller, Ruling the Root: Internet Governance and the Taming of Cyberspace (MIT Press, 2002)

Lawrence Lessig, Code and Other Laws of Cyberspace (Basic Books, 2000) paperback

Lawrence Lessig, The Future of Ideas: The Fate of the Commons in a Connected World (Vintage, 2002) paperback.

Jessica Litman, Digital Copyright: Protecting Intellectual Property on the Internet (Prometheus, 2001)

Philip Jenkins, Beyond Tolerance: Child Pornography Online (NYU Press, 2001)

Shanti Kalathil and Taylor Boas, Open Networks, Closed Regimes: The Impact of the Internet on Authoritarian Rule (Carnegie Endowment for International Peace, 2003)

Supplemental Reading:

For those who want to know more about the technology of the Internet, pick up copies of the various books by Simson Garfinkel or Gene Spafford. For those who want a primer in legal issues, look at www.nolo.com.

Weekly Topics and Reading Assignments:

1. September 2: Course Introduction

2. September 9: Digital Divide and Americans With Disabilities Act

400 students: read Amanda Lenhart et al., "The Ever-Shifting Internet Population," The Pew Internet & American Life Project, April 2003; National Council on Disability, The Accessible Future Washington, DC, June 21, 2001; National Telecommunications and Information Administration, "Falling Through the Net: Toward Digital Inclusion" (October 2000). This report and the three previous reports in this series can be found online at <http://www.ntia.doc.gov/ntiahome/digitaldivide/index.html>

590 students: read Lenhart et al.; National Council on Disability study; NTIA 200 study; some readings of your choice from Benjamin Compaine, ed. The Digital Divide (MIT Press, 2001) or other sources of your choice on the Digital Divide as it relates to the Internet.

3. September 16: History of the Internet

400 students: read Abbate Ch. 1-4, 6

590 students: read Abbate entire book

(If you do not wish to buy a copy, the IU libraries has a copy available online that you can download.)

4. September 23: Laws, Norms, Markets, and Architectures

400 students: read Lessig, Code ch. 1-8, 15

590 students: read Lessig, Code entire book

5. September 30: Governance, Standards, Economics, and Innovation

400 students: read Mueller ch. 1-9

590 students: read Mueller entire book

6. October 7: Spam, Worms, Viruses, and Distributed Denial of Service

400 students: Brian Krebs, "A Short History of Computer Viruses and Attacks", [washingtonpost.com](http://www.washingtonpost.com), 14 February 2003; Edward H. Freeman, "Prosecution of Computer Virus Authors," Legally Speaking, March/April 2003, pp. 5-9; Statement of Eric Holder Before the Subcommittee on Crime, "Internet Denial of Service Attacks and the Federal Response," February 29, 2000

590 students: Krebs; Freeman; Holder; plus any one of the following three: John Eisinger, "Script Kiddies Beware: The Long Arm of U.S. Jurisdiction to Prescribe," [Washington & Lee Law Review](#) 59 (Fall 2002); readings in Kevin Mitnick's book on hacking; or do some online web browsing and bring some recent stories on this subject to class.

7. October 14: Intellectual Property 1: Patents and Trademark Issues (Business Process Patents, Cybersquatting, etc.)

400 students: read Litman ch. 1-5

590 students: Litman ch. 1-5; Janell Kurtz and

Cynthia Mehoves, "Whose Name is it Anyway?" [Marketing Management](#) Jan/Feb 2002, pp.30-34.

8. October 21: Intellectual property 2: Copyright Issues (Peer-to-peer music and video networks, copyleft, creative commons)

400 students: read Litman ch. 6-13,

590 students: read Litman ch. 6-13; in addition, do one of the following two tasks: look online at material about both the GNU General Public License (copyleft) and Creative Commons; or read some parts in Siva Vaidhyanathan, *Copyrights and Copywrongs: The Rise of Intellectual Property and How it Threatens Creativity* (New York University Press, 2001).

9. October 28: guest lecture by Prof. Christine Ogan (Informatics and Journalism) –Internet confession sites. Readings, if any, to be determined.

10. November 4: Digital Government and E-voting

400 students: read Kevin Coleman, "Internet Voting" (CRS Report for Congress, Updated January 31, 2003, Congressional Research Service, Library of Congress); Rebecca Mercuri, "A Better Ballot Box?", *IEEE Spectrum* October 2002, pp. 46-50.

590 students: read Coleman; Mercuri; plus "Voting: Not Yet on the Net", *Minnesota Planning* November 2002, pp. 1-10; Rebecca Mercuri's website

11. November 11: Free speech, Pornography, Cyberstalking, and Gambling

400 students: read Jenkins entire book

590 students: read Jenkins entire book; U.S. General Accounting Office, "Internet Gambling: An Overview of the Issues" (GAO-03-89. December 2002); "Cyberstalking –A New Challenge for Law Enforcement", Ch. 1 in Department of Justice, *Stalking and Domestic Violence Report to Congress*, May 2001; Mark C. Alexander, "The First Amendment and the Problems of Political Viability: The Case of Internet Pornography", *Harvard Journal of Law and Public Policy* vol. 25, pp. 979-1030.

12. November 18: E-commerce and E-taxation

400 students: read Lessig, *Future* ch. 1-8

590 students: read Lessig, *Future* entire book

13. November 25: International Jurisdiction

400 students: read Kalathil and Boas, entire book

590 students: read Kalathil and Boas, entire book; Jack L. Goldsmith, "Against Cyberanarchy," *University of Chicago Law Review* vol. 65 (1998), pp. 1199-1250; David G. Post, "Against 'Against Cyberanarchy,'" *Berkeley Technology Law Journal* vol. 17 (2002), pp. 1365-1387.

14. December 2: guest lecture, Prof. Susan Herring (SLIS), Internet, gender, and communication –class this week runs 5:00 –7:00 PM instead of 4:00 –7:00 PM. Readings to be determined.

15. December 9: Privacy

400 and 590 students: read some in the following sources: Robert Ellis Smith, *Ben Franklin's Web Site: Privacy and Curiosity from Plymouth Rock to the Internet* (Providence, RI: Privacy Journal, 2000); Phil Agre and Marc Rotenberg, ed., *Technology and Privacy: The New Landscape* (MIT Press, 1998); Fred H. Cate, *Privacy in the Information Age* (Washington, DC: Brookings Institution, November 1997).

Assignments

Classroom presentations

Each I400 student will be asked to make a 15-minute classroom presentation on a topic assigned by the instructor. Each I590 student will be asked to make two 15-minute presentations on a topic assigned by the instructor.

Weekly written assignments

Each week (except for weeks 1, 9, and 14) the student is expected to prepare a single question for in-class discussion that week. The top 10 of 12 scores will be counted as part of the final grade. The kinds of questions and how to prepare them will be discussed in week 1. The questions may not be more than one page in length. They must be in the instructor's hands by noon on class day, sent electronically to waspray@indiana.edu. Papers not in my electronic possession by noon will not be accepted. Grading will be as follows:

- 0 - no paper or late paper or unacceptable quality
- 1 - question is on topic but shows limited insight or contains errors
- 2 - question is entirely acceptable but not selected for class use
- 3 - question is accepted for class use but modified by instructor
- 4 - question is accepted for class use without modification

Final assignment

I400 students will have an in-class, open-book, open-note essay exam at the regularly scheduled final exam time.

I590 students will have to prepare a five-page essay due at the time of the regularly scheduled final exam. Details about content will be given later in the semester.

Class Discussion

Every student is expected to come to class having read the assigned material carefully and ready to participate in the discussion. Class participation will be factored into the final grade.

17. Informatics 400/590: Privacy, Security, and Information Warfare

William Aspray
Indiana University in Bloomington

William Aspray is Rudy Professor of Informatics at Indiana University in Bloomington. He also holds faculty appointments in the computer science department, the history and philosophy of science department, and the School of Library and Information Science. He has served as Associate Director of the Charles Babbage Institute for the History of Information Processing, director of the IEEE Center for the History of Electrical Engineering, and executive director of Computing Research Association. He has also taught at Harvard, Penn, Rutgers, Virginia Tech, and Williams. He serves on the history committees of ACM, IFIP, and SIAM. Current scholarship is focused on policy and historical issues relating to information technology. Author's address: School of Informatics, 901 E. Tenth St., Indiana University, Bloomington, IN 47405 USA. Email: waspray@indiana.edu.

Informatics 400/590: Privacy, Security, and Information Warfare (Spring 2004)

Course Description

This course will examine issues of computer privacy, computer security, and information warfare. The focus will mostly be on the Internet, but attention will also be given to databases and to systems such as the telephone and electrical networks that are highly reliant on information technology. While the course will not neglect technical issues, the focus will primarily be on political, social, economic, legal, managerial, and regulatory issues. Topics that we will likely cover include authentication, filtering, encryption, biometrics, cookies, hacking and cracking, employer surveillance, spamming, denial of service, malicious code, cybersquatting, social engineering, the concept of trust, government intrusion versus individual liberty, computer fraud, eavesdropping, identity theft, organizational risk management, national critical infrastructures, and economic espionage. This is a reading and discussion seminar, with some classroom presentations by students and the instructor.

Books to buy for this course –These books are not being ordered by the bookstore, so you are required to order them for yourself. Remember that it may take several weeks for an order to be delivered by mail, so order early enough that you have the books in time for class.

Fred Cate, [Privacy in the Information Age](#) (1997)

Dorothy Denning, [Information Warfare and Security](#) (1998)

Amitai Etzioni, [The Limits of Privacy](#) (2000)

John Hennessy, David Patterson, and Herbert Lin, eds., [Information Technology for Counterterrorism: Immediate Actions and Future Possibilities](#). Computer Science and Telecommunications Board (2003) [This book can be downloaded free of charge in HTML format from: http://www7.nationalacademies.org/cstb/pub_counterterrorism.html]

Stephen Kent and Lynette Miller, eds., [Who Goes There? Authentication Through the Lens of Privacy](#), Computer Science and Telecommunications Board (2003) [This book can be downloaded free of charge in HTML format from: http://www7.nationalacademies.org/cstb/pub_authentication.html]

Steven Levy, [Crypto: How the Code Rebels Beat the Government Saving Privacy in the Digital Age](#) (2002)

Jonathan Littman, [The Watchman](#) (1997)

Kevin Mitnick, [The Art of Deception: Controlling the Human Element of Security](#) (2002)

Stewart Personick and Cynthia Patterson, eds., [Critical Information Infrastructure Protection and the Law](#), Computer Science and Telecommunications Board (2003) [This book may be viewed and printed, one page at a time, in the OpenBook format at: http://www7.nationalacademies.org/cstb/pub_ciip.html. It is also possible to buy a copy or buy a PDF electronic copy.]

Fred Schneider, Trust in Cyberspace, Computer Science and Telecommunications Board (1999) [This book can be downloaded free of charge in HTML format from: http://www7.nationalacademies.org/cstb/pub_trust.html]

Robert Ellis Smith, Ben Franklin's Web Site: Privacy and Curiosity from Plymouth Rock to the Internet (2000)

Weekly Topics and Reading Assignments

Each class will be divided into two parts, A and B, with a break in between. Part A will typically last just less than two hours and will be devoted to discussion of the main readings of the week. Part B will typically last just less than one hour and will be devoted to examination of some technical issue.

Required reading is to be completed prior to the class for which it is assigned. You should read it carefully enough that you can actively participate in the class discussion.

I reserve the right to change the schedule at any time during the semester.

1. January 14 - Introduction.

Part A, B: Class discussion of what is privacy, security, information warfare; structure of the course; ways of theorizing (artifacts have politics, code and other ways of regulation, who makes law); how to approach assigned readings; appropriate questions for your weekly assignments; assignment of Part B presentations to students.

Required reading: none.

Optional reading: Computer Science and Telecommunications Board, The Internet Under Crisis Conditions: Learning From September 11 (2002) [You can view this book in OpenBook format, free of charge but only one page at a time, at: http://www7.nationalacademies.org/cstb/pub_internet911.html]

2. January 21 - Law and policy issues of privacy: an overview

Part A: video on privacy: Jonathan Zittrain, Molly von Houweling, Stanford Law School, 2003.

Part B: Discussion of current events: privacy and automobiles; Northwest data to NASA; *Hiibel v. Nevada*; RFIDs at Walmart, Marks and Spenser

Required reading: none.

Optional reading: David Bruggeman on security, Najma Yousefi on privacy [You will find these chapters as PDF files on Oncourse. They are chapters from the instructor's forthcoming edited book entitled Chasing Moore's Law: Information Technology Policy in the United States.]

3. January 28 - History of Privacy in America.

Part A: Breakout discussions of red, white, and blue groups, followed by general discussion of Smith. Presentation on RFIDs and loyalty cards.

Part B: Student presentation: video surveillance, keystroke monitoring, and Van Eck receptors.

Required reading: Smith (entire book). Pay particular attention to how law, norms, market, and architecture are used to regulate behavior.

4. February 4 - Privacy in an Online World.

Part A: Overview presentation on privacy policy in America: brief history, government as player in policy. Breakout discussions of European and US privacy policies, then general discussion of Cate.

Part B: Student presentation: cookies and privacy.

Required reading: Cate (entire book) - the most important parts are the chapters on European and US policy and on cyberspace (through p. 132). Skip or skim the rest.

5. February 11 - Privacy –A Communitarian View.

Part A: General discussion of Etzioni. We will focus on the four case studies other than the one on strong encryption. Presentation on electronic voting and privacy.

Part B: Student presentation: identity theft.

Required reading: Etzioni (entire book). Feel free to only skim the chapter on strong encryption to get Etzioni's general position since we will be covering this topic in greater depth the following week. Only skim the final chapter quickly. Read the other four case studies and the introductory chapter more carefully.

6. February 18 - Encryption

Part A: Graduate student will lead the general discussion of Levy and make presentation on privacy issues related to digital rights management.

Part B: Student presentation: public key encryption and other kinds of encryption used today.

Required reading: Levy (entire book)

7. February 25 - Introduction to Information Warfare.

Part A: General Discussion of Denning and Litmann. Taped lecture on technologies that lead to new digital rights schema that raise privacy issues.

Part B: Student presentation: digital signatures and watermarks.

Required reading: Denning, pp. 1-76; Littman (entire book)

8. March 3 - Offensive Information Warfare

Part A: Presentation on airline passenger profiling and EU-US disputes over data protection and individual privacy.

Part B: Student will lead general discussion of Denning and make a presentation on privacy issues related to Microsoft's Paladium.

Required reading: Denning, pp 77-282.

9. March 10 - Social Engineering

Part A: General discussion of Mitnick. Class exercise to do some social engineering thought experiments.

Part B: Student presentation: viruses, worms, time bombs. Trojan horses, and other forms of malicious code.

Required reading: Mitnick (entire book).

Optional reading: For a critique of Mitnick, see Jonathan Littman, The Fugitive Game.

March 17 - NO CLASS (SPRING BREAK)

10. March 24 - Defensive Information Warfare

Part A: Student presentation: buffer overflows, password crackers, and other means of hacking.

Part B: General discussion of Denning and Dourish.

Part C: Student presentation: distributed denial of service attacks.

Required reading: Denning, pp. 283-424.

11. March 31 - Authentication

Part A: Student presentation: biometrics and related means of authentication.

Part B: Guest presentation by Prof. Nicole Mineo (IUPUI): HIPAA and medical privacy

Part C: (if time permits, if not, this discussion will occur the following week) General discussion of Kent and Miller.

Required reading: Kent and Miller. While the entire book is assigned, the most important chapters are chapter 1 (especially the long example of authentication), chapter 4 (especially the material about user-centered design), chapter 5 (authentication technologies), and chapter 6 (especially the recent policy examples). Chapter 3 is an excellent introduction to privacy, but you will know all that material by the time you get to this point in the semester.

12. April 7 - Trustworthy Networked Information Systems

Part A: Presentation by Mark Bruhn, VP of Information Security, IU on Defensive Information Warfare at IU. General discussion of Schneider.

Part B: Student presentation: firewalls.

Required reading: Schneider: Chapters 1 through 5. Of particular importance is Chapter 2 in which the trustworthiness of the telephone and Internet networks are compared. Chapter 3 is mostly about software engineering; you do not need to understand all the technical details in that chapter, but do read enough that you understand the approach used by the software engineering community for building software that meets given specifications. Chapter 4 covers some new material and some topics we have already covered (e.g. public key encryption); feel free to skip topics we have already read about for class. Chapter 5 is about building trustworthy systems from untrustworthy components; as with Chapter 3, you don't need to know all the details, but read enough of Chapter 5 to understand the approach.

13. April 14 - Critical Information Infrastructures –Legal and business issues

Part A: General discussion of Personick and Patterson. Presentation on USA Patriot Act.

Part B: Student presentation: spam and pornography filters.

Required reading: Schneider (chapter 6); Personick and Patterson (entire book - it's short!)

14. April 21 - IT and Counter-terrorism

FINAL ASSIGNMENT DUE TODAY!

Part A: General discussion of Hennessy, Patterson, and Lin. Discussion of answers to final assignment.

Part B: Student presentation: preventing theft of service such as Blue boxes, calling card misuse, cellular fraud, and wardriving.

Required reading: Hennessy, Patterson, and Lin (entire book -its short! Pay particular attention to their lists of research projects that need to be done, as described in chapter 3 and in box 5.1)

15. April 28 - open for catch-up, new topics

Classroom presentations

Each I400 student will be asked to join with one other I400 student to make one 45-minute classroom presentation on a technical topic assigned by the instructor. These presentations should include a lecture/tutorial portion. It is also acceptable, but not required, for the presenters to have the other students do some exercises based upon the material presented, organize discussion sections, have mock trials, or do some other activity in which the rest of the class is actively engaged. Assignments will be made at the first meeting of the class.

Each I590 student will be asked to make two presentations on topics assigned by the instructor. Some of these may be technical (Part B) topics; others may involve presenting some material or opening discussion in Part A of the class.

Weekly written assignments

Each week (except for weeks 1, 2, and 15) every student (from both I400 and I590) is expected to prepare a single question for in-class discussion that week. The top 10 of 12 scores will be counted as part of the final grade. The kinds of questions and how to prepare them will be discussed in week 1. The questions may not be more than one page in length. They must be in the instructor's hands by 10:00 AM on class day, sent electronically to waspray@indiana.edu.

Papers not in my electronic possession by this deadline will not be accepted. Grading will be as follows:

- 0 - no paper or late paper or unacceptable quality
- 1 - question is on topic but shows limited insight or contains errors
- 2 - question is entirely acceptable but not selected for class use
- 3 - question is accepted for class use but modified by instructor
- 4 - question is accepted for class use without modification

Final assignment

I400 students will have a short take-home final examination. Details will be given part way through the semester.

The I590 students will work with the instructor as a team to write a research paper on a topic chosen by the instructor on the social, political, legal, or economic dimensions of privacy, security, and information warfare.

Class discussion

Every student is expected to come to class having read the assigned material and ready to participate in the discussion. Class participation will be factored into the final grade.

Religious holidays and cheating

The university has approved policies about missing class and making up work due to absence for religious holidays, as well as unacceptability of cheating and penalties for it. These policies are posted on the IU-B web pages and will be followed in this class.

18. HSSC120: Computer Learning for the Arts and Sciences

Nathan L. Ensmenger
University of Pennsylvania

Nathan Ensmenger is an assistant professor in the Department of the History and Sociology of Science at the University of Pennsylvania. His current research focuses on the intersection of work, computers, and organizations. *Author's address:* 303 Logan Hall, University of Pennsylvania, 249 S. 36th Street, Philadelphia, PA 19104 USA.
Email: nathanen@sas.upenn.edu.

HSSC120: Computer Learning for the Arts and Sciences

Nathan L. Ensmenger
University of Pennsylvania

Course Description

Information technology is a ubiquitous component of the modern university. Understanding what computers are, how they work, and how they can best be used for research and presentation is an essential element of a well-rounded undergraduate education in any discipline. The objective of this course is to provide students in the School of Arts and Science with a broad perspective on the social, economic, and technological developments associated with the so-called “Information Revolution.” The overall focus will be on the fundamental concepts that transcend any particular technology; the goal is to provide students with the knowledge needed to readily and skillfully adapt to a rapidly changing technological environment. An added benefit is that they will also acquire some skills that will be relevant to them in other coursework and possibly even their future careers.

Although no prior exposure to computing (or higher mathematics) will be required, students will have the opportunity to become familiar with the fundamentals of programming and computer science. More importantly, however, they will learn how to better evaluate the role of computing in their own lives, research, and careers.

Section I: Computers and How They Work

Week I: When Computers Were People ...

Readings

Martin Campbell-Kelly and William Aspray. *Computer: A History of the Information Machine*.
Read Chapters 1, 2.

Week II: The Origins of Electronic Computing

Readings

Alan Bierman, *Great Ideas in Computer Science* (1997). Read Chapter 7 (“Electronic Circuits”).

Week III: The Universal Machine

Readings

John Von Neumann, First Draft of the Report on EDVAC (1945)

Visit Andrew Hodges’ web site devoted to the Alan Turing. Read the documents describing the Turing Machine as a Universal Computer, and explore the Virtual Turing Machine. The site can be found at the following URL:

<http://www.turing.org.uk/turing/scrapbook/machine.html>.

You should be prepared to describe in a few paragraphs what a Turing Machine does and why it is significant.

Week IV: New Modes of Computing

Readings

Campbell-Kelly/Aspray, *Computer*. Chapters 8-9.

Section II: Introduction to Programming

Week V: The Science of Software

Readings

Bierman, Chapter 8 ("Machine Architectures").
Lab Handout 1: Hello, World!

Week VI: The Art of Programming

Readings

Biermann, Chapter 3 ("Numerical Computation")
Donald Knuth, "Computer programming as an art," reprinted in *ACM Turing Award Lectures: The First Twenty Years* (New York: ACM Press, 1987), pp. 33-46.
Lab Handout 2: Loops

Week VII: Artificial Reality?

Readings

John Conway's Game of Life:
www.bitsorm.org/gameoflife
Visit the site and explore the Java-based Game of Life simulation.

Lab Handout 3: Disease Models and Simulations

Week VIII: Programming in the Real World

Readings

Timothy Lethbridge, "What Knowledge is Important to a Software Professional? " in Gilda Pour, Martin Griss, and Michael Lutz, "The Push to Make Software Engineering Respectable," *Computer* 33, 5 (2000), 35-51.
Lab Handout 4: Bugs, Glitches, and Other Errors

Section III: Computers and Society

Week XIX: You Say You Want a Revolution?

Readings

Campbell-Kelly/Aspray, Chapter 9 (pages 207-229)
Sherry Turkle, "Hackers: Loving the Machine for Itself" from *The Second Self: Computers and the Human Spirit* (New York, NY: Simon and Schuster, 1984) –Chapter 6.

Week X: Internet I –How it all works

Readings

Vannevar Bush, "As We May Think"

<http://www.isg.sfu.ca/ensuremath{\sim}duchier/misc/vbush/vbush-all.shtml>

Janet Abbate, *Inventing the Internet*. Selected chapters.

Week XI: Internet II –What it all means

Readings

Frank Webster, "Introduction" and "Information and the Idea of an Information Society"
Chapters 1 and 2 of *Theories of the Information Society*, (1995).

Langdon Winner, "Mythinformation," in *The Whale and the Reactor*, University of Chicago Press, 1986.

Week XII: Programming for the Internet

Readings

Alex Pang, Old Wine for New Bottles: Making the Britannica CD Multimedia Timelines.
<http://www.hb.se/bhs/ith/1-99/askp.htm>

Explore the following site on web design:

<http://www.webpagesthatsuck.com/>

Online reference on Composing Good HTML

<http://www.ology.org/tilt/cgh/>

Lab Handout 5: HTML

Week XIII: Virtual Utopias and the Cyber Society

Readings

Richard Coyne, *Technoromanticism: Digital Narrative, Holism, and the Romance of the Real* (MIT Press, 1999). Chapter 1, "Digital Utopias."

Richard Stallman, "Why Software Should Be Free."

<http://www.gnu.ai.mit.edu/philosophy/shouldbefree.html>

Turkle, Sherry "Virtuality and its Discontents: Searching for Community in Cyberspace," *The American Prospect* no. 24 (Winter 1996): 50-57

<http://epn.org/prospect/24/24turk.html>

Lab Handout 6: The Do-It-Yourself Crypto-Anarchist

Week XIV: Big Finish

19. ITEC-1210: IT Revolution-Myth or Reality?

Kim Fortun
Rensselaer Polytechnic Institute

Kim Fortun is the associate dean of the School of Humanities and Social Sciences, and associate professor in the Department of Science and Technology Studies at Rensselaer. She is also the Director of Rensselaer's Center for the Ethics of Complex Systems. She received her doctorate in cultural anthropology at Rice University, and is the author of *Advocacy after Bhopal: Environmentalism, Disaster, New Global Orders* (University of Chicago Press, 2001). Her current work includes the study of information technology, environmental justice, and social stratification. *Author's address:* Department of Science and Technology Studies, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 USA. Email: fortun@rpi.edu.

INFORMATION TECHNOLOGY REVOLUTION: MYTH OR REALITY?

IHSS 86849-86854 / 1210.01-1210.06 ITEC 86822-86827 / 1210.01-1210.06

FALL 1999, Monday-Wednesday 2:00-3:50

Instructors:

Kim Fortun: email: fortun@rpi.edu	1210.01 phone: x2199	Monday: Sage 3303 office: Sage 5408	Wednesday: Eaton 215 office hours: M/W 1-2
Todd Cherkasky email: cherkt@rpi.edu	1201.02 phone: x6381	Monday: Sage 3303 office: Sage 5502	Wednesday: Lally 104 office hours: M/W 1-2
Ron Eglash email:	1201.03 phone: x2048	Monday: Darrin 337 office: Sage 5114	Wednesday: Lally 102 office hours: M/W 12:30-1:30
Torin Monahan email: monaht@rpi.edu	1201.04 phone: x8503	Monday: Darrin 337 office: Sage 5704	Wednesday: Sage 3101 office hours: M 12-1:30
Atsushi Akera email: akeraa@rpi.edu	1201.05 phone: x2314	Monday: Darrin 330 office: Sage 5206	Wednesday: Lally 02 office hours: M/W 1-2
Virginia Eubanks email: eubanv@rpi.edu	1201.06 phone: x8503	Monday: Darrin 330 office: Sage 5704	Wednesday: LOW 4034 office hours: W 12-2

** all instructors are also available by appointment, if you can't attend their office hours **

How do IT revolutions happen??? How is IT innovated?? What do IT students today need to learn to become innovators themselves? These questions orient this course. We will explore how innovation of IT has happened so far, and techniques that will fuel creative development of IT in the future.

The course can be thought of as a "Grand Exploration" of IT -- from alphabets, the printing press and the first world maps to GIS and GPS; from telephones and radio to cybernetics, the gene chip and bioinformatics; and --of course -- from the first computer to the Internet. One goal is for you to acquire the investigative skills necessary to assess how technological innovation happens, and affects the world. Another goal is for you to learn about the history of IT, so that you can use the past to "think the future."

You will experiment with different research techniques, including library and web research, interviewing and data analysis. One important assignment will involve an interview with an IT researcher at Rensselaer. Another important assignment will be to "profile" an information technology, mapping the context of its emergence and its wide-reaching impact. Together, students' profile projects will provide a complex map that links any given information technology to other technologies, and to broad social, cultural and political-economic trends. Peer-reviews of both the interview and the profile project will help you learn to constructively critique other people's work, and to take advantage of feedback you receive to improve your own work. Group presentations of your interview material and profile projects will provide experience working in teams. Your two exams will provide the opportunity to demonstrate how the research skills you have learned can help you analyze *any* information technology -- preparing you for a world in which new information technologies will constantly emerge and deserve rigorous analysis.

COURSE MATERIALS AND MEETINGS

All readings for the course are available electronically –through either the Internet or Rensselaer Library’s electronic reserve system –which you can get to by going to the RPIInfo webpage <<http://www.rpi.edu/rpinfo>>, clicking on “Electronic Reserves” and going to the course identified as IHSS 1210. Note that we did not choose the assigned readings because they contain uncontested truths. They are important *interpretations* of history, shaped by each author’s particular perspective. Thus, ALL the articles should be read critically, and comparatively. Important questions you should ask as you read include the following: What is the basic chronology outlined in the article? What does the chronology emphasize? The role of certain social actors? Political forces? Social or economic effects? What does the chronology leave out?

Some weeks, your homework includes a film. Films will be screened on Thursday evenings in DCC 330, and then will be put on reserve in the library. We strongly recommend trying to see the films on Thursdays. It will be much easier than trying to get the film over the weekend, and it will give you the opportunity to jump-start your work for the next week –with your classmates. In week 5, your homework will include the playing of the computer game SimEarth, which you can purchase through the bookstore.

All readings, films and other assignments for each week should be completed *prior* to the class meeting for which they are listed.

The class will meet in a number of different venues: Most Mondays you will meet with your particular class for a lecture and discussion. Most Wednesdays you will meet with a smaller group of students and your section leader for discussion and interactive exercises. Most weeks your section will also meet electronically, in discussions on WebCT. And many weeks there will be a film screened on Thursday evenings –you can either attend the screening, or check out the film over the weekend from the library. On occasion, all three classes of the course will meet together. You’ll be notified in class and by email when the joint-meetings will occur. All joint-meetings will be held in Sage 3303.

In each Monday session of the course our goal will be to profile a particular IT innovation. You need to come to class prepared to participate in this. First we’ll do a “synchronic analysis” that suggests the particular social and cultural tenor of the time in which the innovation being profiled emerged, or became widely used. We’ll ask what else was going on, and what would have gone into a time capsule of the period. Then we’ll map at least 10 things that came together to make the emergence of this IT innovation possible: social forces, economic factors, culture, technology, etc. Then we’ll map at least 10 after-effects, asking how this IT innovation entered and changed the world.

On most Wednesdays, we’ll think of the “after-effects” as “catalysts,” so that we can explore opportunities and needs for IT innovation in the future.

A third venue for this class will be within the bulletin board option of WebCT, where we can have an on-line discussion. To do this, you need to create an account in WebCT for this course –by going to the RPIInfo webpage <<http://www.rpi.edu/rpinfo>> then clicking on “WebCT Courses.” Please record your password so that you don’t forget it. To participate in a bulletin board discussion, you first click on the “bulletin board” option on the Welcome Page. Then click on “forums” on the left tool bar. This should bring up an index that lists each week, and each section. Go to your section, in the appropriate week. There you will find a list of questions related to that week’s reading. You need to contribute a response to at least one of the questions, building on the responses by other students already posted. Each response should be at least 200 words long and should make at least one direct reference to the assigned reading. Your contribution to WebCT discussion needs to be posted by Monday mornings at 8:00.

COURSE PROCEDURES

academic honesty

Throughout the course, you are encouraged to work collaboratively with your peers. Discussion of course material, both in and outside class, is appropriate throughout the term –except during the in-class exams. Note that *all* parties to cheating are accountable. If you allow another student to borrow your exam answers you, too, are guilty of academic dishonesty. If you allow your written assignments to be copied and submitted by other students, you, too, will be held responsible.

In all written work, citations must be included for both indirect and direct quotation, providing clear documentation of sources. Note that we may request submission of all your sources so that we can check the accuracy of your citations.

It is not acceptable to resubmit work done for another course unless you have received explicit permission from your section leader. Representation of work done by another person as your own work is also unacceptable.

Please see the *Rensselaer Student Handbook* for complete guidelines on academic honesty. Note that penalties for plagiarism can be very harsh. **Specifically: if we are able to confirm plagiarism or cheating on *any* written assignment in the course you are likely to fail *the entire course*.**

electronic citizenship

In this course, as in all activities at Rensselaer, you are expected to abide by the Institute's "Policy on Electronic Citizenship," which articulates how "the ethical principles that apply to everyday academic community life also apply to the use of information and computing resources." Please familiarize yourself with the details of this policy, available at <<http://www.rpi.edu/web/comec/>>.

gender-neutral language

Contemporary standards of language use encourage gender-neutral language. Male-gender pronouns or words like "man" to refer to everyone are not empirically accurate. Thus, in this course please use gender-neutral language. See the tips provided by the Writing Center at <<http://www.rpi.edu/dept/lc/writecenter/web/text/gender.html>>.

team work

Your course projects will be developed and presented in teams. You are expected to participate in the development of ideas for the presentation, to attend scheduled group meetings to plan the presentation, to complete your share of the preparation on schedule, and to produce high quality material for the class presentation itself. Failure to do your share of the team's work will be judged very harshly. Evaluation of team collaboration will be carried out by each team member, and will affect your final course grade.

grade appeals

You may appeal a grade through a written statement describing the grounds on which a change of grade is considered appropriate. The written statement should reference one of the three criteria identified in the *Student Handbook* as grounds for appeal. According to the *Handbook*, the "allegation must be based upon a violation of the course syllabus, a violation of Institute policy, or a violation of the student's rights under the Student Bill of Rights."

GRADING

Grade evaluation for the course will be based on the following percentages:

PARTICIPATION (CLASS + WEBCT)	25%
FIRST EXAM	15%
SECOND EXAM	15%
INTERVIEW WITH IT RESEARCHER	15%
PROFILE PROJECT AND CLASS PRESENTATION	30%

(A=85-100%; B=70-84%; C=55-69%; D=40-54%)

attendance

Class attendance is required. Three unexcused absences will lead to failure of the course. Absences will be considered excused if make-up work is completed, and the absence was due to illness, a family emergency, or other similar situations. Heavy workload in other classes does not constitute grounds for an excused absence. To receive an excused absence, you must complete all regular assignments due the day of your absences –within one week of the absence. You also must submit a 400 word essay that uses assigned readings to describe some aspect of the “profile” focused on that week. The extra essay also must be submitted within one week of your absence. Valid reasons for extending this deadline for submitting materials related to an excused absence may be discussed with your section leader.

participation in class and WebCT discussion

Your participation in class and WebCT discussions will be graded and count as 25% of your grade. Each of ten WebCT posts will count as 1 point (1%). You will receive either 1 point, or none at all, without the opportunity to resubmit. Your section leader will inform you via email to your WebCT mailbox if you did not receive credit. If you did not receive credit, you need to take greater care in preparing your next post – making sure it responds to the questions being discussed, and is substantiated with references to assigned readings. You can build a total of 10 points toward your final grade through your WebCT posts.

Your section leader will be responsible for giving you a grade for participation in class discussions and exercises. Failure to submit benchmarking materials –like your choice of an IT researcher to interview, or a topic for your profile project –will affect your class participation grade. You can build a total of 15 points toward your final grade for your class participation.

exams

You will have two exams, each of which will count as 15% of your final course grade. The first exam will be on Wednesday, October 13. The second exam will be on Monday, November 15. Make-up exams will not be given except in cases involving extreme illness or family emergencies.

Both exams will include multiple choice questions, short answer questions and essay questions. In the multiple choice and short answer components of the exam, you will be asked to demonstrate that you have done the reading for the course, and listened well to lectures and class discussion. In the essay component, you will be asked to demonstrate that you have learned *theories and methods* for understanding the innovation and effects of IT: You may be asked to provide a complex analysis of one of the technologies profiled in class. You also may be asked to demonstrate how the course has prepared you to analyze *any* IT innovation. In other words, the essay component of the exam will ask you to show that you have learned to use the theories and methods focused on in the course well enough to be able to use them in the future –in other class assignments, and in your endeavors beyond the classroom.

interview with IT researcher at Rensselaer

This assignment involves choosing an IT researcher at Rensselaer who you would like to interview so that you can learn more about what she or he does as a researcher, what resources are necessary to conduct the research, and what results the research is expected to produce. To choose your interviewee, visit the IT web page, where IT researchers are listed according to the topic of their research; often there are links to researcher’s own web pages.

This assignment is an opportunity for you to explore possible second disciplines –and to help others understand the different domains of work within IT. It will also help you develop important research skills. Throughout your career you will need to learn about what others do and how it relates to what you are interested in. Good interviewing skills are crucial.

Two students should not interview the same researcher. There will be a section in the bulletin board of WebCT where you can post who you plan to interview. Whoever posts first has priority. If there are more students in the class than IT researchers on campus, we will discuss alternative guidelines in class.

You should have selected an interviewee by September 29. You should have conducted the interview by October 27. The complete write-up of the interview is due November 17. When you turn in the complete

write-up of the interview you also must turn in email confirmation that your interviewee has checked her or his quotes for accuracy. You also must turn in two “peer reviews” confirming that at least two of your classmates have read your write-up for clarity and comprehensiveness. The write-up should be approximately five double-spaced pages.

Your interview material will be presented in class November 29 & December 1 in groups organized around possible second disciplines or other thematic foci. It is your responsibility to make sure you are in a group.

The written component of the interview assignment is worth 10 points and the class presentation is worth 5 points. In all, you can build 15 points toward your final course grade through this assignment.

profile projects

The most extensive assignment in this course is to “profile” an information technology, following the protocol provided in class. The challenge is to map out the world in which an information technology emerged, what led up to it, and what came after. Each of the twenty categories on the profile protocol should be responded to in approximately 300 words. If some categories require more than 300 words, others may be less. In all, you should compose an essay of approximately 7000 words (20-25 pages), including an introduction.

Two students should not profile the same technology. There will be a section in the bulletin board of WebCT where you can post your preference. Whoever posts first has priority. Be creative. Last year, students chose technologies ranging from the Julian calendar to the Pony Express, FedEx, cell phones, satellites –and even the basketball shot clock.

Your profile is due in increments. The first step is to submit a synchronic analysis of the time at which the technology you are studying was invented or became popularly used –due October 6. Then you will begin submitting sections of the profile. Five sections are due October 20; another five sections are due October 27; another five sections are due November 3; the last five sections are due November 10. Each time you submit a set of five sections you must turn in “peer reviews” to confirm that at least two of your classmates have edited your work for clarity and comprehensiveness. Each set of five sections also must include two bibliographic references, properly cited. Whenever you turn in a set of sections, please include a profile schema that highlights which sections are included in a given set.

The final paper -- including twenty sections, an introduction and a profile schema –is due November 29.

The projects will be presented in groups of five on December 6 & 8. Each group will have fifteen minutes to present. We will take time in class to organize groups for presentations, but you must work together outside of class to put the presentations together.

The profile project contributes 30 points (30%) to your final grade. The sections submitted October 20-November 10 are each worth 1 point –for a possible total of 20 points. Another five points can be built through the final submission of the paper. Another five points can be built through the class presentations.

You can raise the grade you receive on the first draft of your sections if you go to the Writing Center and receive a stamp indicating that they have reviewed the revisions made for the final version of the paper. Thus, the final version of the paper can involve substantial re-writing, with substantial improvements of your grade.

Further details on the profile projects will be posted in WebCT.

SEMESTER OVERVIEW, WITH DUE DATES

WEEK ONE	<u>USING THE PAST TO THINK THE FUTURE</u>	
Monday, August 30:		
Wednesday, September 1:	biosketch	
WEEK TWO:	<u>FROM ALPHABETS TO LITERACY TO SOCIAL INFORMATICS</u>	
Wednesday, September 8:	WebCT post	
WEEK THREE	<u>MAPPING THE WORLD, CIRCA 1500/CIRCA 2000</u>	
Monday, September 13:	WebCT post selection of topic for profile project	
Wednesday, September 15:		
WEEK FOUR	<u>FROM CANALS TO A WIRED RPI: TROY, NY AS IT HUB</u>	
Monday, September 20:	WebCT post	[joint meeting / Sage 3303]
Wednesday, September 22:		
WEEK FIVE	<u>FROM TELEPHONES TO TELE-CULTURE</u>	
Monday, September 27:	WebCT post	
Wednesday, September 29:	selection of IT researcher to interview	
WEEK SIX	<u>RADIO: JUMPSTARTING VIRTUAL LIVING</u>	
Monday, October 4:	WebCT post	[joint meeting / Sage 3303]
Wednesday, October 6:	synchronic analysis for profile project	
WEEK SEVEN	<u>EXAMINING IT REVOLUTIONS</u>	
Wednesday, October 13:	exam	
WEEK EIGHT	<u>FROM CYBERNETICS TO CHAOS TO COMPLEXITY</u>	
Monday, October 18:	WebCT post	[joint meeting / Sage 3303]
Wednesday, October 20:	first quarter of profile project with two peer reviews	
WEEK NINE	<u>MAINFRAMING MACHINES AND SOCIETIES</u>	
Monday, October 25:	WebCT post	
Wednesday, October 27:	second quarter of profile project with two peer reviews confirmation that interview has been done	
WEEK TEN	<u>CODING LIFE, BUILDING BIOTECH: COMPUTERS+BIOLOGY+ECONOMICS</u>	
Monday, November 1:	WebCT post	[joint meeting / Sage 3303]
Wednesday, November 3:	third quarter of profile project with two peer reviews	
WEEK ELEVEN	<u>GAMING MICROPROCESSORS</u>	
Monday, November 8:	WebCT post	
Wednesday, November 10:	fourth quarter of profile project with two peer reviews	
WEEK TWELVE	<u>PROJECTING THE INTERNET</u>	
Monday, November 15:	WebCT post	[joint meeting / Sage 3303]
Wednesday, November 17:	interview write-up with 2 peer reviews	
WEEK THIRTEEN	<u>RE-EXAMINING IT REVOLUTIONS</u>	
Monday, November 22:	exam	
WEEK FOURTEEN	<u>DESIGNING IT: THE NEW EXPERTS</u>	
Monday, November 29:	profile project group presentations of interviews + team work evaluation	
Wednesday, December 1:	group presentations of interviews + team work evaluation	
WEEK FIFTEEN	<u>MAPPING INFORMATION TECHNOLOGY REVOLUTIONS</u>	
Monday, December 6:	group presentations of profile projects + team work evaluation	
Wednesday, December 8:	group presentations of profile projects + team work evaluation	

WEEK ONE USING THE PAST TO THINK THE FUTURE

(Monday, August 30 / Wednesday, September 1)

PREPARATION

reading

Drucker, Peter

"The Next Information Revolution," p47-58 *Forbes ASAP*. August 24, 1998. [7p]

[To read this, you'll need to learn to access Rensselaer Library's electronic reserves]

webbing

Time Capsule Website <<http://abcnews.go.com/century/timecapsule/index.html>>

WebCT Student Self-Study Guide <<http://webct.rpi.edu:8900/public/dss>>

[Create an account for yourself. Login. Print out the self-study guide. Then go through the online exercises –so that you are ready to use WebCT by Week 2.]

DUE

Wednesday biosketch

WEEK TWO: FROM ALPHABETS TO LITERACY TO SOCIAL INFORMATICS

(Wednesday, September 8) (HOLIDAY: Monday, September 6)

PREPARATION

watching

Excerpts from Modern Times and Metropolis [60 minutes]

reading

Chandra Mukerji & Bruce Jones, "Introduction" + "Four Important Periods in the History of the Book" + "The Development of Print Technology," *Manuscripts, Books and Maps: The Printing Press and a Changing World*

<<http://communication.ucsd.edu/bjones/Books/>> [6p]

"Playboy Interview: Marshall McLuhan –A Candid Conversation with the High Priest of Popcult and Metaphysician of Media," *Playboy* March 1969. [reprinted as p233-269 in *Essential McLuhan*, 1995] [37p]

Rob Kling, "What is Social Informatics and Why Does it Matter?" *D-Lib Magazine*, January 1999.

<<http://www.dlib.org:80/dlib/january99/kling/01kling.html>> [read p1-3 & p7-22] [18p]

Sherry Turkle, "Seeing Through Computers: Education in a Culture of Simulation," *The American Prospect*, March-April 1997 <<http://epn.org/prospect/31/31turkf.html>> [13p]

webbing

Ralph Lombreglia, "What Happened to Multi-Media?" *Atlantic Unbound* June 5, 1997.

<<http://www.theatlantic.com/unbound/digicult/dc9706/dc9706.htm>>

DUE

Wednesday WebCT post

WEEK THREE MAPPING THE WORLD, CIRCA 1500/CIRCA 2000

(Monday, September 13 / Wednesday, September 15)

PREPARATION

watching

The Spice Route: The Discovery of the Sea Lane to Africa and Asia (by Ebbo Demant) [90 minutes]
The Invisible Shape of Things Past & Global Mapping (Art + Com / Joachim Sauter/ 1996) [22 Minutes]

reading

Chandra Mukerji & Bruce Jones, "Maps and Views of the New World" *Manuscripts, Books and Maps: The Printing Press and a Changing World* <<http://communication.ucsd.edu/bjones/Books/>> [3p]

Neil Stephensen (1996) "Mother Earth Mother Board," *Wired Magazine* December 1996
<<http://www.wired.com/wired/archive/4.12/ffglass.html>> [7p + skim]
[Read the first seven pages carefully, then skim the rest.]

Walton, Anthony, "Technology Versus African Americans," *Atlantic Monthly* January 1999.
<<http://www.theatlantic.com/issues/99jan/aftech.htm>> [8p]

webbing

Latitude Website [designed by Pat Seed]

<<http://www.ruf.rice.edu/~feegi/>>

[Spend substantial time on this site, since the technology we will be profiling is "latitude." Make sure to explore what cartography is, what cartography allowed people in the 16th century to do, and how cartographers today use Geographic Information Systems (GIS)].

History of Cartography

<<http://www.geosys.com/cgi-bin/genobject/cartography/tig5e6>>

Map Viewer: world 0.00N 0.00E [designed by Xerox Parc]

<<http://pubweb.parc.xerox.com/map>>

MapInfo

<<http://www.mapinfo.com/>>

[This is the site for a very successful company started by a Rensselaer alum. Students in *Politics and Economics of IT* in the spring semester will use MapInfo software in their course projects].

Cook Inlet Keeper / Homer, Alaska [GIS for watershed data]

<<http://www.xyz.net/~keeper/>>

recommended

Henrikson, Alan

"The Power and Politics of Maps," p49-70 *Reordering the World: Geopolitical Perspectives on the 21st Century* edited by George Demko and William Wood. Boulder: Westview, 1994. [20p]

DUE

Monday WebCT post

Wednesday selection of topic for profile project

WEEK FOUR *FROM CANALS TO NETWORK LOGIC TO A WIRED RPI: TROY, NY AS IT HUB*

(Monday, September 20/ Wednesday, September 22)

PREPARATION

watching

Troy, New York

reading

Chandra Mukerji & Bruce Jones, "The Rise of the University," *Manuscripts, Books and Maps: The Printing Press and a Changing World* <<http://communication.ucsd.edu/bjones/Books/>> [read only p1] [1p]

P. Thomas Carroll (1999) "Designing Modern America in the Silicon Valley of the 19th Century," p22-27 *Rensselaer Alumni Magazine*. March [3p]

Bruce Adams (1999) "That's IT! The Wonderful World of Information Technology, p16-21 *Rensselaer Alumni Magazine*. March [3p]

Press Release: Rensselaer Ranked 5th most Wired College
<http://www.rpi.edu/dept/NewsComm/Renss_news/press_releases/yahoo99.htm> [1p]

Todd Oppenheimer, "The Computer Delusion," *The Atlantic Monthly* July 1997.
<<http://www.theatlantic.com/atlantic/issues/97jul/computer.htm>> [26p]

Betsy Wagner, Stephen Gregory, Richard Bierck, Missy Daniel & Jonathan Sapers, "Computers Do Work" p83-93 *US News and World Report* December 2, 1996. [11p]

Susan Gregory Thomas, "Software that turns kids on" p97-98 *US News and World Report* December 2, 1996. [2 p]

Stephen Brier, "In the Digital Universe, Learning Comes Alive" <<http://www.ashp.cuny.edu/sbrier.html>> [5p]

Paul Starr, "Computing Our Way to Educational Reform," *The American Prospect* July-August 1996.
<<http://epn.org/prospect/27/27star.html>> [18p]

webbing

IT Research at Rensselaer

<<http://www.rpi.edu/IT/research.html>>

Project Links

<<http://links.math.rpi.edu/>>

DUE

Monday WebCT post

Wednesday synchronic analysis for profile project

WEEK FIVE FROM TELEPHONES TO TELE-CULTURE

(Monday, September 27 / Wednesday, September 29)

PREPARATION

watching

The Telephone: A Quest for Instant Communication (50 min)

reading

Sam Pitroda, "Development, Democracy and the Village Telephone" p66-79 *Harvard Business Review* November-December 1993. [14p]

Avital Ronell "Watson, I Need You," p247-255 *Major Problems in the History of Technology* edited by Merritt Roe Smith and Gregory Clancey. Houghton Mifflin, 1998. [9p]

Katherine Smith "Memoir of a Telephone Operator, 1930" p236-240 *Major Problems in the History of Technology* edited by Merritt Roe Smith and Gregory Clancey. Houghton Mifflin, 1998. [5p]

Venus Green "Personal Service in the Bell System," p255-263 *Major Problems in the History of Technology* edited by Merritt Roe Smith and Gregory Clancey. Houghton Mifflin, 1998. [8p]

Bruno Latour "Ma Bell's Road Trip," p263-266 *Major Problems in the History of Technology* edited by Merritt Roe Smith and Gregory Clancey. Houghton Mifflin, 1998 [4p]

Howard Rheingold, "Look Who's Talking: The Amish," *Wired Magazine*. January 1999.
<<http://www.wired.com/wired/archive/7.01/amish.html>>

Mark Cooper, "Universal Service: A Historical Perspective and Policies for the 21st Century" [read first section, titled: "Universal Service: A Century of Commitment"]
<<http://www.benton.org/Library/Prospects/commitment.html>> [6p]

Joseph Pelton, "Telecommunications for the 21st Century," p80-85 *Scientific American*. April 1998. [5p]

webbing

"The Big Schmooze: How to Give Cell Phone," *Wired Magazine*. January 1999.
<<http://www.wired.com/wired/archive/7.01/cellphone.html>>

The Benton Foundation
<<http://www.benton.org>>

recommended

Tom Farley, "Telephone History: Parts 1, 2, & 3"
<<http://www.privateline.com/TelephoneHistory/History1.htm>>

DUE

Monday WebCT post
Wednesday selection of IT researcher to interview

WEEK SIX RADIO: JUMPSTARTING VIRTUAL LIVING

(Monday, October 4 / Wednesday, October 6)

PREPARATION

watching

Empire of the Air: The Men Who Made Radio (120 min)

reading

Susan Douglas "Amateur Operators and American Broadcasters: Shaping the Future of Radio," p364-371 *Major Problems in the History of Technology* edited by Merritt Roe Smith and Gregory Clancey. Houghton Mifflin, 1998 (1986). [8p]

Erik Barnouw, "Forebears," p3-14 *Tube of Plenty: the Evolution of American Television*, 1975. [12p]

David Bianculli, "Instant Replay: A Broad Look at Broadcast History" p41-62 *Teleliteracy*, 1992. [22p]

David Hughes and Dwayne Hendricks, "Spread-Spectrum Radio," p94-96 *Scientific American* April 1998. [3p]

Todd Lappin, "Deja Vu All Over Again," *Wired Magazine* May 1995.
<http://www.wired.com/wired/archive/3.05/dejavu_pr.html> [9p]

webbing

100 Years of Radio Web

<<http://www.alpcom.it/hamradio/>>

Radio's War of the Worlds Broadcast (1938)

<<http://members.aol.com/jeff1070/wotw.html>>

Microradio Empowerment Coalition

<<http://www.radio4all.org/>>

Grassroots Radio from Harare, Zimbabwe.

<<http://www.freespeech.org/rbo/>>

Radio Free Europe

<<http://www.theatlantic.com/unbound/citation/wc961225.htm>>

DUE

Monday WebCT post

Wednesday synchronic analysis for profile project

WEEK SEVEN EXAMINING IT REVOLUTIONS

(Wednesday, October 13) (HOLIDAY: Monday, October 11)

PREPARATION

!!!EXAM REVIEW!!!

WEEK EIGHT FROM CYBERNETICS TO CHAOS TO COMPLEXITY

(Monday, October 18 / Wednesday, October 20)

PREPARATION

watching

Fast, Cheap and Out of Control [120 minutes]

reading

Ron Englash "Cybernetics and American Youth Subculture," *Cultural Studies*. [on reserve] [23p]

Kevin Kelly (1994) "Emergence of Control," p111-127 *Out of Control: The New Biology of Machines, Social Systems and the Economic World*. [on reserve] [17p]

Frijof Capra (1993) "A Systems Approach to the Emerging Paradigm," p230-237 *The New Paradigm in Business: Emerging Strategies for Leadership and Organizational Change* edited by Michael ray and Alan Rinzler. [on reserve] [8p]

webbing:

SimEarth (computer game)

recommended

F. Heylighen, C. Joslyn & V. Turchin, "What are Cybernetics and Systems Science,"
<<http://pespmc1.vub.ac.be/:/CYBSWHAT.html>>

DUE

Monday WebCT post

Wednesday first quarter of profile project with two peer reviews

WEEK NINE MAINFRAMING MACHINES AND SOCIETIES

(Monday, October 25/ Wednesday, October 27)

[4S]

PREPARATION

reading

Martin Campbell-Kelly and William Aspray, "The Computer Becomes a Business Machine," chapt 5 *A History of the Information Machine*. (26p) (on reserve)

W. Wayt Gibbs, "Taking Computers to Task," *Scientific American* July 1997.

<<http://www.sciam.com/0797issue/0797trends.html>>

Asaf Goldschmidt and Atsushi Akera, "John Mauchly and the Development of the ENIAC Computer," An Exhibition in the Department of Special Collections, Van Pelt Library, University of Pennsylvania.

<<http://www.library.upenn.edu/special/gallery/mauchly/jwmintro.html>>

webbing

"A History of Computers," <http://www.maxmon.com/hmain.htm>

Environmental Defense Fund: Scorecard

<<http://www.scorecard.org/>>

DUE

Monday WebCT post

Wednesday second quarter of profile project with two peer reviews
confirmation that interview has been done

WEEK TEN CODING LIFE, BUILDING BIOTECH: COMPUTERS+BIOLOGY+ECONOMICS

(Monday, November 1 / Wednesday, November 3)

PREPARATION

watching

Decoding the Book of Life [60 minutes]

The Gene Hunters [60 minutes]

reading

Cassandra Franklin-Barbajosa, "DNA Profiling: The New Science of Identity," *National Geographic* May 1992.

[on reserve] [10p]

Charles Platt, "Evolution Revolution: By 2005, the Human Genome Project will have transcribed the entire programming language of human life," *Wired Magazine* January 1997.

<http://www.wired.com/wired/archive/5.01/ffgenome_pr.html> [12p]

Kristen Philipkoski, "Identifying Genetic Standards," *Wired News* October 22, 1999.

<<http://www.wired.com/news/print/1,1294,32040,00.html>> [2p]

Zina Moukheiber, "Chip + DNA = hype" p126-127 *Forbes* June 15, 1998.

[on reserve] [1p]

Vicki Glaser & John Hodgson, "Before Anyone Knew the Future Nature of Biotechnology" p239-242
Nature Biotechnology Vol. 16 March 1998.

[on reserve] [4p]

Dan Seligman, "Outlawing DNA" p110-115 *Forbes*, July 6, 1998.

[on reserve] [3p]

William Wells, "Biotech Revolutions: Selling DNA Data," Access Excellence Website

<<http://www.accessexcellence.org/AB/BA/genomics/index.html>> [5p]

[read introductory page, then click on and read article titled "Selling DNA Data."]

Morrison Institute for Population and Resource Studies, "Human Genome Diversity Project: Frequently Asked Questions"

<<http://www.stanford.edu/group/morrinst/hgdp/faq.html>> [14p]

Alaska Native Knowledge Network, "Declaration of Indigenous Peoples of the Western Hemisphere Regarding the Human Genome Diversity Project"

<<http://www.ankn.uaf.edu/declaration.html>> [2p]

webbing

Incyte Pharmaceuticals, Inc. "The Future of Drug Research, Point-and-Click Biology, etc.,"

<<http://www.Incyte.com/products/lifeseq.html>>

The National Human Genome Research Institute

<<http://www.nhgri.nih.gov>>

recommended

Evelyn Fox Keller, "The Body of a New Machine: Situating the Organism Between Telegraphs and Computers" p79-118 *Refiguring Life: Metaphors of Twentieth-Century Biology*. [40p]

DUE

Monday WebCT post

Wednesday third quarter of profile project

WEEK ELEVEN GAMING MICROPROCESSORS

(Monday, November 8 / Wednesday, November 10)

PREPARATION

reading

Martin Campbell-Kelly and William Aspray, "The Shaping of the Personal Computer," p476-486 *Major Problems in the History of Technology* edited by Merritt Roe Smith and Gregory Clancey. Houghton Mifflin, 1998. [excerpted p236-257 *Computer: A History of the Information Machine*. Basic Books, 1996] [22p] [on reserve]

Ted Nelson, "Computer Lib," p473-476 *Major Problems in the History of Technology* edited by Merritt Roe Smith and Gregory Clancey. Houghton Mifflin, 1998. [first published in 1974] [4p] [on reserve]

Sherry Turkle, "The Second Self," p486-495 *Major Problems in the History of Technology* edited by Merritt Roe Smith and Gregory Clancey. Houghton Mifflin, 1998. [excerpted p182-192 *The Second Self: Computers and the Human Spirit*. Simon & Schuster: 1984.] [11p] [on reserve]

Jim Carlton, "They Coulda Been a Contender: The full, inside story of Apple's biggest, most strategic blunder," *Wired Magazine*, November 1997. [15p]
<http://www.wired.com/wired/archive/5.11/es_apple.html>

Michael Dertouzos, "Creating the People's Computer," *Technology Review*, April 1997. [10p]
<<http://www.techreview.com/articles/apr97/dertouzos.html>>

Charles Mann, "Programs to the People," *Technology Review*, January 1999 [10p]
<<http://www.techreview.com/articles/jan99/mann.htm>>

webbing

"Chronology of Events in the History of Microcomputers" by Ken Polsson.
<<http://www3.islandnet.com/~kpolsson/comphist.htm>>

"The High Cost of High Tech" & "The Environmental Costs of Computers" & "The Dark Side of High Tech" by Corporate Watch
<<http://www.corpwatch.org/trac/feature/hitech/index.html>>

Computer Professionals for Social Responsibility
<<http://www.cpsr.org>>

recommended

Eric Raymond, "The Cathedral and the Bazaar"
<<http://www.tuxedo.org/esr/writings/cathedral-bazaar/>>

DUE

Monday WebCT post
Wednesday third quarter of profile project with two peer reviews

WEEK TWELVE *PROJECTING THE INTERNET*

(Monday, November 15 / Wednesday, November 17)

PREPARATION

reading

Howard Rheingold, "Visionaries and Convergences: The Accidental History of the Net," p65-109 *The Virtual Community: Homesteading on the Electronic Frontier*, 1993.[45p]
[on reserve]

Audrie Krause, "Selling Cyberspace: The Commercialization of the Internet" [3p]
<<http://www.igc.org/trac/feature/feature1/krause.html>>

U. S. Department of Commerce / National Telecommunications and Information Administration, Government, "Falling Through the Net: Defining the Digital Divide." Third in the series, July 8, 1999. [13pp]

Scan main page: <<http://www.ntia.doc.gov/ntiahome/digitaldivide/index.html>>

Read Part I closely: <<http://www.ntia.doc.gov/ntiahome/ftn99/part1.html>>

Mark Dery, "With Liberty and Justice for Me: Is the Internet giving ordinary people more control over their lives?" *Atlantic Unbound*, July 22, 1999. [8p]
<<http://theatlantic.com/unbound/digicult/dc990722.htm>>

Peter Drucker, "Beyond the Information Revolution," *Atlantic Monthly*, October 1999. [15p]
<<http://www.theatlantic.com/issues/99oct/9910drucker.htm>>

webbing

Hayashi, Alden

"The Net Effect," *Scientific American* January, 1999.

<<http://www.sciam.com/1999/0199issue/0199infocus.html>>

Plugged In

<<http://www.pluggedin.org>>

Silicon Valley Toxics Coalition

<<http://www.svtc.org/resource.htm>>

recommended

"As We May Think," Vannevar Bush, *Atlantic Monthly*, July 1945

<http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm> (20pp)

DUE

Monday WebCT post

Wednesday interview write-up with two peer reviews

WEEK THIRTEEN RE-EXAMINING IT REVOLUTIONS
(Monday, November 22) (HOLIDAY: Wednesday, November 24)

PREPARATION

!! EXAM REVIEW!!

MONDAY: EXAM

WEEK FOURTEEN DESIGNING IT: THE NEW EXPERTS
(Monday, November 29 / Wednesday, December 1)

PREPARATION

!! INTERVIEW PRESENTATIONS !!

DUE

Monday group presentation of interviews + team work evaluation
Wednesday group presentation of interviews + team work evaluation

[the due date for your profile paper has been changed –to Friday, Dec 3]

WEEK FIFTEEN MAPPING INFORMATION TECHNOLOGY REVOLUTIONS
(Monday, December 6 / Wednesday, December 8)

PREPARATION

!! PROFILE PROJECT PRESENTATIONS!!

DUE

Monday group presentations of profile projects + team work evaluation
Wednesday group presentation of profile projects + team work evaluation

PROFILING INFORMATION TECHNOLOGY

[CATALYSTS]

[AFTER-EFFECTS]

→ → → → → → → → → → →

events

events



politics

politics



society

society



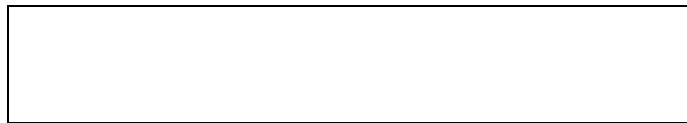
economics

economics



concepts

concepts



institutions

institutions



technologies

technologies



individuals

individuals



culture

culture



contingencies

contingencies

→ → → → → → → → → → →

20. ITEC-1220: The Politics and Economics of IT

Atsushi Akera
Rensselaer Polytechnic Institute

Atsushi Akera is an assistant professor in the Department of Science and Technology Studies at Rensselaer Polytechnic Institute. He is currently working on a book, *Calculating a Natural World: Computers, Scientists and Engineers During the Rise of U.S. Cold War Research*, which uses the history of computing as a lens into the early U.S. institutional infrastructure for cold war research. *Author's address:* Department of Science and Technology Studies, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 USA. Email: akeraa@rpi.edu.

ITEC/IHSS 1220: The Politics and Economics of Information Technology

Tuesday / Friday 12:00 –1:50 pm (DCC 318)

Lead Instructor: Atsushi Akera

Sage 5206, x2314, akeraa@rpi.edu

Office Hours: F 2-3p*

Lane DeNicola Sage 5703, x6171 denicl@rpi.edu ofc hrs: F 2-3p*	Noelle Foster Sage 5709, x2551 fosten2@rpi.edu ofc hrs: F 10a-12n	Sean Lawson Sage 5705, x2711 lawsos@rpi.edu ofc hrs: F 2-3p*	Selma Sabanovic Sage 5703, x6171 sabans@rpi.edu ofc hrs: F 2-3p*	Alex Sokoloff Sage 5703, x6171 sokolof@rpi.edu ofc hrs: T 3-4p*
---	--	---	---	--

*office hours also by appointment

Course Description:

What is information technology (IT) worth?? Are the ideals of freedom and economic prosperity compatible? What is the difference between capitalism and democracy? Do the latest information technologies change the balance between these two fundamental institutions of our society? Who benefits, and who gets left behind?

This course offers you a Faustian bargain. You will be given an opportunity to be the CEO of a mock IT startup, and undertake IT design efforts with the assistance of other students in the class. We will teach you how to “read” the practical politics and economic opportunities presented by new information technologies. The course will also help you design more effective IT systems, and may bolster your career as a computer or IT professional. In return, you will be asked to explore the world of social advocacy and multi-cultural awareness and responsibility...

Overview of Course Requirements:

The major requirements for this course are the following. Your grade may also be modified for reasons associated with the required readings, attendance, extra credit, class participation and team effort. The instructors also reserve the right to modify a grade based on individual circumstances.

- Team component:
 - Project Proposals and Public Debates 25%
 - Final Advocacy Project 25%
- Individual component: 50%
 - Weekly Short Essays
 - Position Papers
 - Field research of an IT user or stakeholder (Optional)

Required Texts: (The books are available at the Rensselaer Union Bookstore)

- Lawrence Lessig, *Code and Other Laws of Cyberspace* (Basic Books, 1999)
- Andrew Herman and Thomas Swiss (eds.), *World Wide Web* (Routledge, 2000)
- *P&E of IT Course Manual* (<http://www.rpi.edu/~akeraa/pe-IT>)
- Online articles and websites (")

Please read this syllabus and the *P&E of IT Course Manual* carefully. They are your “bible” for this course! The course is based on utilizing the pedagogic technique of large-scale entrepreneurial simulation, and hence we will be expecting professional conduct in every aspect of this course.

The Scenario:

Here is the basic scenario for the course. A new philanthropic foundation, *The IT Futures Foundation*, was established four years ago in Silicon Valley. This foundation is committed to the idea that new information technologies can contribute to social equity, global awareness, user-friendly systems, and the strength of democratic institutions. The foundation is willing to entertain proposals from progressive firms and non-profit organizations. It is prepared to commit about \$100 million a year in new venture capital. This group of philanthropists is also committed to bringing professional management practices to philanthropic organizations. As a consequence, they are asking all firms to approach them with well-developed business plans and technology development proposals. The foundation will also be sponsoring a series of public debates on major issues regarding IT and society.

Establishing an IT Venture:

At the outset of this course, we will ask for approximately fifteen (15) volunteers to serve as the CEO of new IT startups and non-profit organizations with a commitment to social advocacy. You are welcome to ask your friends and acquaintances to join you in this venture. During Week 3, we will hold a “jobs fair” where you can “hire” other students into your firm or non-profit organization.

Team Projects:

A detailed description of all team projects may be found in the *P&E of IT Course Manual* posted on the course website. In general, all team members will be engaged in three different types of exercises as part of the team effort:

- A public debate on a current issue about IT and society
- Regular project proposals submitted in response to a Request for Proposal (an “RFP”) from the IT Futures Foundation
- A final advocacy project, usually based on developing one of the regular proposals into a full-length (15-30 page) proposal. This project has other components and is submitted in multiple drafts.

Individual Work and Assigned Readings:

We’ve designed this course to let you choose where to place your efforts, and to work from your strengths. This means, however, that you will have to do **some work beyond the required readings**.

Required Readings: Read all of the “required readings” materials posted on the “Course Readings and Schedule” page of the course website. Also spend 10-30 minutes at each website listed as a “webbing” assignment. You must complete the required readings by the first class meeting of the week for which the readings are assigned.

Weekly Short Essay Questions: Each week, you will also be asked to write a brief essay in response to two questions posted on the “Course Readings and Schedule.” This is not meant to be a difficult assignment. Your answers, combined, may be from 300-500 words in length. Your essays must be original, and indicate that you have done at least all of the required reading and webbing assignments. (You must refer to all of the required materials in some way.)

Essays should be typed, double spaced. Grades for the short essay assignment are issued on a modified pass/fail basis. You have up to two weeks to resubmit an essay that received a grade of Qualified Pass or Not-Yet-Pass. (Exception: all revisions must be turned in by the last day of

class.) Submit both your revision **and your original essay** containing the instructor's comments. All essays are due in class on Tuesdays (Friday if the class does not meet on Tuesday).

Each paper that receives a "Pass" or "Pass+":	2 pt
Each paper that receives a "Qualified Pass":	1 pt

Position Papers: During the semester, you will have four opportunities to turn in either a new draft, or a revised draft, of a position paper. You may turn in only one paper on or before each of the four deadlines (see "Course Schedule" below for the deadlines). Think of this as a way of writing up the work you do in preparing for a debate or a response to an "RFP," and getting individual credit for the work you do. You may also research a topic independently and submit it as a paper. You can find a list of suggested position paper topics in the *P&E of IT Course Manual* posted on the course website. We will evaluate these position papers as if they were submitted to a professional newsletter, *The Future of IT*, on the social aspects of IT.

Superior:	12-20 pts
Accepted:	10 pts*
Revise and Resubmit:	5 pts*
Rejected:	0-4 pts

Please note that in most instances, position papers will earn at most a "Revise and Resubmit" on the first draft. "Superior" papers, in particular, are supposed to be semester-long research papers that must be submitted in multiple drafts. If you wish to work on such a paper, consult with your section instructor early in the semester. (*You may resubmit any paper marked "Revise and Resubmit" or "Accepted, Revision Recommended") In general, papers that qualify for "revise and resubmit" must be based on 4-6 substantial sources beyond the required readings, and should be between 1,200 and 1,500 words in length. "Accepted" papers should be correspondingly larger, and seriously engage with all recommended improvements given by your section instructor. Your paper will be evaluated based on the dual criteria of the amount of effort and the quality and originality of your analysis.

When submitting a position paper, indicate whether you would like feedback on the mechanics of your writing (language, grammar, style, organization, etc.) and whether you are willing to have your paper appear in the course newsletter.

Example: W:No / NL:Yes

Position papers must include complete citations. See "Academic Citation Guidelines" in the *P&E of IT Course Manual* for details.

Field Research of an IT User or Stakeholder (optional): Your team project must be based on user-centered design, which must include field research involving potential users and other stakeholders. Individual team members may interview up to two users or stakeholders. Other types of field research projects may also be negotiated with the instructor. See *P&E of IT Course Manual* for details.

Each interview, with confirmation and revision: 0-7 pts.

Extra Credit (optional): The instructors may announce extra credit lectures and other assignments worth 1 pt. each.

Grades and Evaluation:

Individual Component: The individual component of your grade will be based on the following scale, except as modified as stated elsewhere in the syllabus (see especially “Class Participation” below):

Points:	(see below)	30	22	16	<16
Grade:	A	B	C	D	F

“A”s will be awarded primarily on the basis of the quality, not quantity, of your work. This can include how well you incorporate the recommended and/or additional readings into your weekly essays (often reflected by a “Pass +”) and position papers. Individual grades are assigned at the sole discretion of your section instructor.

Team Component: The team component of your grade is based on the grade your team earns, but it may be modified by **up to plus one or minus three letter grades** based on your contributions to the team. The final judgment about any adjustment is made by your section instructor, based on your self-evaluation, the evaluation of your team members, and a confidential evaluation submitted by your CEO.

Team Grade for Project Proposals & Debates: Your team’s grade for the project proposals and debates will be based on the following scale:

Points:	(see below)	24	16	8	<8
Grade:	A	B	C	D	F

“A”s are awarded primarily on the basis of the quality of your team’s work and preparations, and not the total points accumulated. This component of the team grade will be determined through a conference of all section instructors. Given that this is an academic exercise, we will emphasize the respect and constructive comments you offer other teams, as opposed to excessive competition.

Team Grade for Final Advocacy Project: Your section instructor will assign a grade for the final advocacy project based on the overall quality of all components of the project. The quality of the final draft of the proposal will be a significant part of the grade, but earlier preparations, drafts, peer critiques (if required by your instructor), field research (interviews), and responsiveness to instructor feedback will also be considered. The final presentation is also an important component of your team’s final project grade.

Individual								
Short Essays		Position Papers		Field Research		Extra Credit		
							Total Points:	
Class Participation	High (A)	–	Moderate	–	Low (F)	+1 to - 1 Grade	Indiv. Grade:	

Team								
RFP&Debate pts		RFP&D Grade		Final Project:		Team Grade (Unadj):		
Individual Effort	High (A)	–	Moderate	–	Low (F)	+1 to - 3 Grade	Team Grade: (adjusted)	

Final:								
Individual Grade		Team Grade		Absences w/o makeup		Adj.	Final Grade:	

Assigned Readings and Course Schedule:

The full list of assigned readings, websites, and assignments schedule may be found at the course website, <http://www.rpi.edu/~akeraa/pe-IT>. (Select: "Syllabus & Course Materials" >then> "Readings/Schedule")

Course Schedule Summary:

The IT Futures Foundation will schedule public debates and presentation of regular project proposals during most Fridays in plenary. However, please pay attention to the RFP announcements as some of the deadlines will differ. Position papers deadlines are scheduled for Friday on the stated weeks (Tuesday if there is no class on Friday). Additional deadlines and events are summarized below.

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Proposal				X	X		X		X	X		X		X	
Debate						X		X	X		X		X	X	
Position Paper					X		X				X				X
Other Dates:	E		J		S	A _{pp}			FR	A ₁		A ₂		FR _c	Ev/A _F

Key to Specific Deadlines and Events:

Symbol:	Date:	Event / Assignment Due:
E	January 13 & 16	Establish Entrepreneurial Firms
J	January 30	IT Jobs Fair
S	February 13	Stakeholder Analysis
A _{pp}	February 20	Preliminary Proposal for Final Advocacy Project,
FR	March 16	Field Research (Interview) Write Up
A ₁	March 23	Advocacy Project (first full draft)
A ₂	April 9	Advocacy Project (second full draft)
FR _c	April 23	Field Research Confirmations and Revisions
A _F	April 27	Advocacy Project (final draft) & Class Presentation
Ev	April 27	Self Evaluation Form / CEO's Evaluations

All assignments are **due in class**

The Fine Print:

Attendance: As a matter of policy, attendance is required in all H&SS courses. You may miss two plenary and two section meetings without penalty. Beyond this, you must make up any additional absences, **excused or unexcused**, through extra assignments as negotiated with your section instructor. You are still responsible for turning in all assignments as required by the syllabus. If you miss a scheduled meeting with your section instructor without notifying her or him twenty-four hours in advance (use email), this will also count as an absence (one section meeting). Excessive absences, if not made up, will substantially affect your final grade.

Class Participation: Class participation is very important in this course. Your instructor may modify your individual grade by **up to one letter grade** based on her or his judgment about your class participation, and your contributions to the class as a whole. Attendance does not constitute class participation.

Writing vs. Class Participation: In recognizing that some students have real difficulties speaking up in class, the section instructor, at his or her discretion, may grant individual students the right to submit additional written work, or place greater effort into the normal written assignments in lieu of class participation. Specific arrangements must be made with the instructor at the outset of the course.

Required Readings: We apologize for being somewhat strict on this score, but some common foundation of knowledge is essential for this class to work for everyone. Repeated failure to do the required readings will result in an assessed penalty of one-third of a letter grade (final grade) for each occasion where you are unable to demonstrate familiarity with the required readings.

Late Submissions: Section instructors may set their own policies on late assignments. Unless instructed otherwise, the deadlines in this course are firm. No late papers or projects will be accepted except through specific arrangement with your instructor.

Gender Fair Language: Students in this course are expected to use gender fair language in their writing. Every time you use a masculine-oriented word to refer to people in general, the implicit effect, even if unintended, is to whisper: *women don't count*. Essays that do not use gender fair language will not receive a passing grade. If you are unfamiliar with the practice of gender fair writing, you should read "Gender Fair Language," written by Jenny Redfern of RPI's Writing Center. See, www.rpi.edu/web/writingcenter/genderfair.html.

The Writing Center: Writing is an important component of IT work. Believe it. In addition to proposals and reports, you will be writing five to ten memos and emails each day. Your performance will always be evaluated on how well you convey your ideas. Periodically, you may be advised to seek out the services of the Writing Center. The Writing Center is located in Sage 4508. You may obtain further information at 276-8983, or www.rpi.edu/web/writingcenter/. Your section instructor may also require you to have someone at the Writing Center go over your written assignment before you can resubmit it for consideration. If this is the case, you **must obtain a stamp from the Writing Center**, and then turn in both the stamped and revised version of your assignment. Keep in mind that improving the mechanics of writing on any assignment will not be enough to receive a higher grade if the content remains inadequate.

ESL / LD Students: The requirements for this course will be adjusted to serve the needs and capabilities of ESL and LD students. Students who have difficulties reading or writing should feel free to notify their section instructor about their particular situation. In general, the guideline we will use is to require at least four hours of reading per week. Likewise, ESL/LD student should expect to spend an average of two hours per week on their written assignments, exclusive of teamwork.

Use of Student Generated Materials: All materials submitted as part of a team project (proposals, interviews, critiques, etc.) to your instructor will be treated as public information. It may be posted on the course website, or distributed through other media, for the benefit of other students and instructors.

Academic Honesty: Student-teacher relationships are built on trust. Students must trust that teachers have made appropriate decisions about the structure and content of the course, and teachers must trust that the assignments students turn in are their own. Acts that violate this trust undermine the educational enterprise. They contradict our very reason for being at Rensselaer. The *Rensselaer Handbook* defines various forms of academic dishonesty and the procedures for responding to them.

Students should note, in particular, that the penalties for plagiarism are quite harsh. They can vary from failure of assignment to failure of course, based on the student's intent and the severity of the case. Any use of another person's work, including the use of specific ideas and interpretations, must be acknowledged through proper attribution. All direct use of another person's words must be placed in "quotations." You must indicate when you paraphrase another person's work. For further guidelines, see "Academic Citation Guidelines" in the *P&E of IT Course Manual*. If you have any questions concerning this policy, or any other policy pertaining to academic honesty, please ask for clarification.

Part IV. Historical Case Studies for Computing and Information Systems Education

21. What Was the Question? The Origins of the Theory of Computation

Michael S. Mahoney
Princeton University

Abstract

Sciences work against their historical memory as they reshape the curriculum to conform to the current state of the field. In doing so, they often lose sight of the questions, concerns, and goals that originally motivated the development of the subject. This paper traces the origins of the theory of computation to bring out the diverse range of disciplinary research agendas that converged to form the now central concept of the Chomsky hierarchy of languages and automata.

Note: *The following is a brief overview of a study still under way, portions of which have been published as articles and are accessible via my webpage.¹²⁹ Hyperlinks at the end of paragraphs lead to further discussions in those articles and in drafts not yet published. The main diagram is also available via a web link, and contains links to those discussions.*

Why history? In the sciences (which, for my purposes here, include engineering) the demands of the present and the future work against our memory of the past. To keep pace with growing knowledge, the sciences continually reconstitute themselves, revising the curriculum to conform to the conceptual structure of what we know rather than retracing the path by which we came to know it. To borrow the well-known biological maxim, rather than recapitulate phylogeny, ontogeny ignores it. Hard-won results, often originally unrelated, become corollaries of more general concepts and techniques; what required years of research becomes an exercise left to the reader. Clearly, much is gained in the process, but much is lost, both historically and pedagogically. In emphasizing the logical structure of a field, one loses sight of its historical evolution. One casts as “natural” connections among concepts that were conceived independently of one another, often for quite different purposes. In focusing on the answers those concepts now provide, one loses sight of the questions that initially stimulated their development and, hence, of how the questions have changed.

That loss of historical perspective seems particularly regrettable for computer science, a field that has from the outset had difficulty defining itself, deciding whether it is one science or several or, for that matter, to what extent it is, can, or even should be a science at all.¹³⁰ Hence, while considering the current subject, it seems important to keep in view the interests and goals that motivated the early development of a theory of computation. What were the questions? Why theoretical computer science, and where did it come from?

¹²⁹ www.princeton.edu/~mike/computing.html.

¹³⁰ For some of the issues involved, see my “Software as Science, Science as Software,” in *History of Computing: Software Issues*, ed. Ulf Hashagen et al. (Berlin/Heidelberg/New York, 2002) [<http://www.princeton.edu/%7Emike/softsci.htm>], especially the addendum, written in light of audience response to the paper, and Wolfgang Coy, “Defining Discipline,” in *Foundations of Computer Science: Potential, Theory, Cognition* (Berlin/Heidelberg/New York, 1997). [http://waste.informatik.hu-berlin.de/Coy/Coy_Defining_Discipline.html]

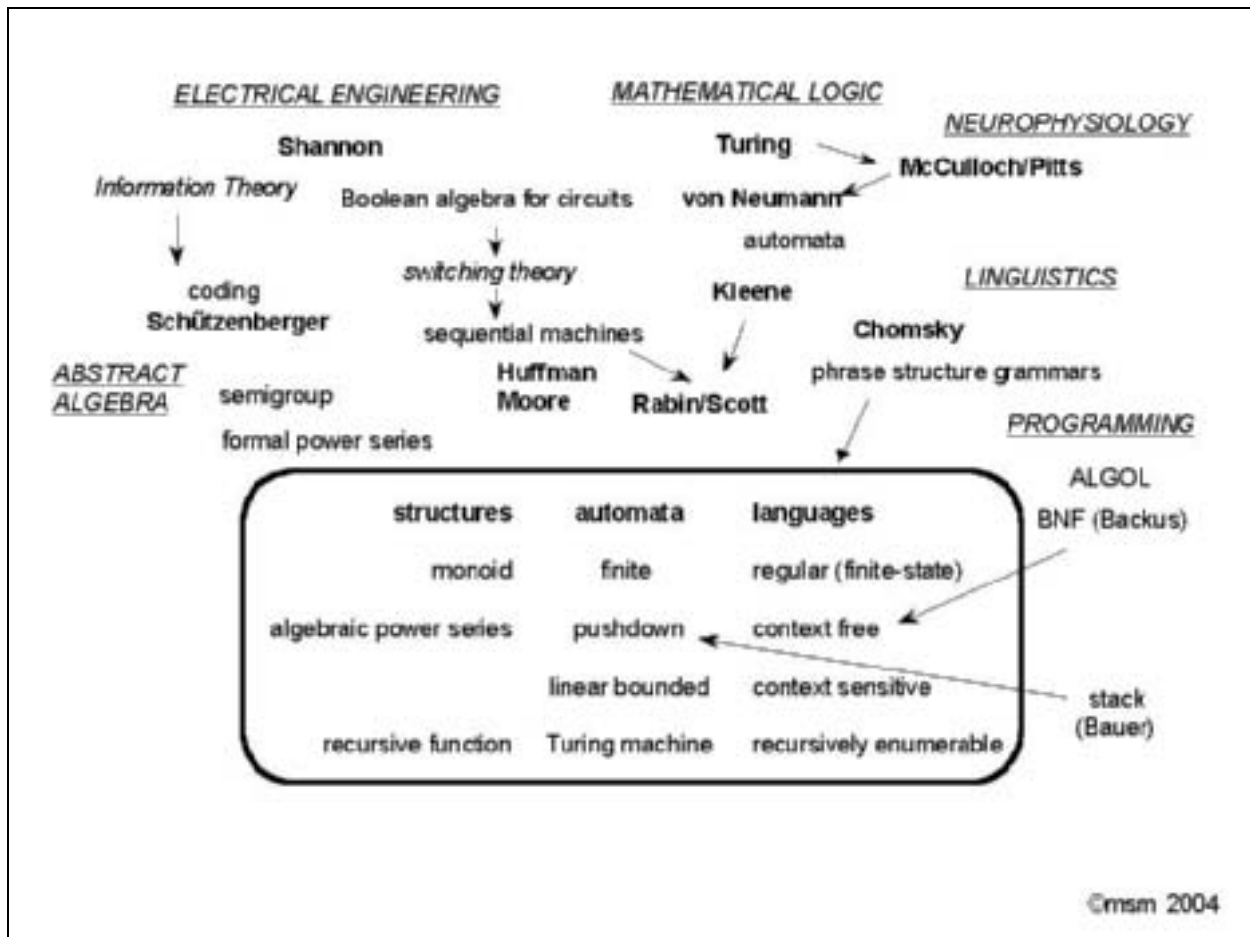
Scientific disciplines, indeed disciplines in general, are perhaps best understood in terms of their agendas—that is, what their practitioners agree is “to be done.” What are the questions to be answered, the problems to be solved? What constitutes answers to those questions and solutions to those problems? What are the means by which they are achieved? How do we know when we have them? One becomes a practitioner of a discipline by learning its agenda and how to carry it out, starting with what is already known and moving to what is not. One achieves standing by solving the problems recognized as hard or important. One exercises leadership in extending the agenda, often through solutions that reveal new and deeper problems.

The notion of agendas emphasizes the communities that share them. A discipline is a group of people formed around a common agenda. That view makes disciplines fluid and changeable. People move in and out of disciplines as their own agendas change. New disciplines form around agendas that no established discipline is willing or able to incorporate into its own. Disagreements among practitioners about the agenda can lead to division into separate disciplines. Overlapping or converging agendas can lead either to the union of disciplines or to the spawning of a new discipline from portions of others. Computer science, especially that portion of it called “theoretical,” is a prime example of a new science born of convergence. Between 1955 and 1970, a new agenda formed around the theory of automata and formal languages, which increasingly came to be viewed as foundational for the field as a whole. Figure 1 is meant to capture the main elements of the process, as it occurred over the decade 1955-65.¹³¹

In the center of the figure is the familiar “Chomsky hierarchy” of phrase-structure languages with their corresponding automata, which forms the core of any course on the theory of computation. Less familiar is the family of corresponding mathematical structures, which constitute the less commonly taught algebraic theory of machines and languages. Surrounding the box are some of the people, concepts, and disciplines that combined to form that set of correspondences. They reflect the initially quite separate agendas of neurophysiology, electrical engineering, mathematical logic, mathematics, linguistics, machine translation, programming, and computer design. At the start, no one was aiming at a unified hierarchy of machines, languages, and mathematical structures. Except for the Turing machine at one end and switching circuits at the other, none of the languages or automata existed as objects of study. Rather, they emerged from conjoint or concurrent efforts in a variety of fields. What follows is a brief overview of the main points of convergence, together with references to the seminal papers.

¹³¹ A similar story can be told about formal semantics. See the pertinent section in my “Computer Science: The Search for a Mathematical Theory,” in *Science in the 20th Century*, ed. John Krige and Dominique Pestre (Amsterdam, 1997), Chap. 31.

Figure 1. Converging Research Agendas 1955-65



Perhaps the best place to begin is with John von Neumann and his “General and Logical Theory of Automata,” first presented at the Hixon Symposium.¹³² There von Neumann expanded on references he had made in the EDVAC report to the work of McCulloch and Pitts on the logical capabilities of nerve nets.¹³³ Von Neumann was intrigued by the cybernetic notion of the computer as an artificial organism or, conversely, of the central nervous system as a “natural automaton.”¹³⁴ The analogy with natural organisms offered the promise of reciprocal benefits. On the one hand, the computer could provide a model of biological processes, including those of the brain. On the other, those processes included self-regulation, self-replication, and evolution, all of which might serve as guides to the fundamental problem of constructing reliable

¹³² “A General and Logical Theory of Automata,” originally published in *Cerebral Mechanisms in Behavior: The Hixon Symposium*, ed. L.A. Jeffries (New York: Wiley, 1951), 1-31; repr. in *Papers of John von Neumann on Computing and Computer Theory*, ed. William Aspray and Arthur Burks (Cambridge, MA, 1987) [hereafter GLTA, with page references to the reprint edition].

¹³³ W.S. McCulloch and W. Pitts, “A Logical Calculus of the Ideas Immanent in Nervous Activity,” *Bulletin of Mathematical Biophysics* 5 (1943): 115-33.

¹³⁴ On cybernetics, see Steve J. Heims, *The Cybernetics Group* (Cambridge, 1991).

devices from unreliable components. Interested for his purposes in what came to be called “growing automata” (cellular automata), von Neumann had little to say about fixed or finite automata as exemplified by the state table of a Turing machine or the circuitry of a computer. He also had little to say about how to go about creating a “theory of automata which deserves that name, that is, a properly mathematical-logical theory,” which he took to be essential for further progress.¹³⁵ [More on von Neumann]¹³⁶

Automata

Following von Neumann’s lead, the study of finite automata emerged from the convergence of lines of research in mathematical logic, neurophysiology, and electrical engineering. Logician Stephen C. Kleene later recalled that, while at RAND in 1951, he was asked to see what he could do with the model of nerve nets set out in the McCulloch-Pitts paper, which he reconsidered “having newly in mind the idea of a finite automaton, which came to me through reading the printer’s proof of von Neumann’s Hixon Symposium lecture.”¹³⁷ The result was his study, “Representation of events in nerve nets and finite automata,” which circulated widely even before its publication in 1956. In it, he established the relation between finite automata and regular events (later regular languages).

At the same time, the electrical engineers had undertaken the extension of Shannon’s Boolean analysis of circuits to sequential circuits, in which state transitions depend on both input and previous states. The theory originated in the early 1950s with two investigations, David A. Huffman’s “The Synthesis of Sequential Switching Circuits” in 1954 and E.F. Moore’s “Gedanken-Experiments on Sequential Machines,” published in 1956 but circulating for some time before that.¹³⁸ The two articles approached sequential machines from opposite directions. Huffman focused on the circuits themselves: given a set of inputs and outputs, how does one construct a sequential circuit that will transform the one into the other? In particular, how does one structure the internal states of the machine to minimize the circuitry needed to realize them? By contrast, Moore concentrated on their external behavior, treating the devices as “black boxes” and asking what could be deduced about their circuitry from their response to sequences of inputs.

In articles written independently and at different times, but published in the same volume, Moore, working from sequential circuits, and Kleene, working from nerve nets, both converged on the abstract model of the finite state machine characterized by its input-output behavior. Precisely because of the finite nature of the device, it seemed closer in structure to the digital computer than did the Turing machine with the potentially infinite memory of its tape, and in the late 1950s the community of people interested in the theory of computation increasingly turned their attention to the simpler device to probe its possibilities and limits. The seminal paper came from two Princeton graduate students engaged in summer research at the IBM Research Center in 1957. In “Finite automata and their decision problems,” Michael O. Rabin and Dana S. Scott spoke directly to the allure of finite automata:

¹³⁵ Von Neumann, GLTA, 405.

¹³⁶ <http://www.princeton.edu/~mike/cra/vonNeumann.html>

¹³⁷ Stephen C. Kleene, “Origins of Recursive Function Theory,” *Annals of the History of Computing* 3/1 (1981), 64.

¹³⁸ Huffman, “The Synthesis of Sequential Circuits,” *Journal of the Franklin Institute* 257/3 (1954): 161-90; Moore, “Gedanken-Experiments on Sequential Machines,” in *Automata Studies*, ed. C.E. Shannon and J. McCarthy (Princeton, 1956), 129-53.

Turing machines are widely considered to be the abstract prototype of digital computers; workers in the field, however, have felt more and more that the notion of a Turing machine is too general to serve as an accurate model of actual computers. It is well known that even for simple calculations it is impossible to give an a priori upper bound on the amount of tape a Turing machine will need for any given computation. It is precisely this feature that renders Turing's concept unrealistic.

In the last few years the idea of a finite automaton has appeared in the literature. These are machines having only a finite number of internal states that can be used for memory and computation. The restriction of finiteness appears to give a better approximation to the idea of a physical machine. Of course, such machines cannot do as much as Turing machines, but the advantage of being able to compute an arbitrary general recursive function is questionable, since very few of these functions come up in practical applications.¹³⁹

In their article, Rabin and Scott joined the work of Kleene (as mediated by an unpublished treatment by John Myhill) with the definition of automata used by Moore to establish what kinds of strings of symbols finite automata can recognize and, thereby, to show that effective procedures exist for determining what a given automaton, viewed as a black box, can do.

At the end of his article, Kleene suggested that "the study of a set of objects a_1, \dots, a_r under a binary relation R , which is at the heart of the above proof [of his main theorem], might profitably draw on some algebraic theory."¹⁴⁰ Rabin and Scott mentioned in passing that the tapes of an automata form under concatenation a free semigroup with unit. But neither of the articles pursued the algebra any further. The algebraic theory came rather from another quarter. In 1956, at the seminal Symposium on Information Theory¹⁴¹ held at MIT, Marcel P. Schützenberger contributed a paper, "On an Application of Semi-Group Methods to Some Problems in Coding." There he showed that the structure of the semigroup captured the essentially sequential nature of codes in transmission, and the semigroup equivalent of the "word problem" for groups expressed the central problem of recognizing distinct subsequences of symbols as they arrive.

The last problem, in turn, is a version of the recognition problem for finite automata as defined and resolved by Kleene and then Rabin and Scott. Turning from codes to automata in a series of papers written between 1958 and 1962, Schützenberger established the semigroup as the fundamental mathematical structure of automata theory, demonstrating that the states of a finite-state machine, viewed as a homomorphic image of the equivalence classes of strings indistinguishable by the machine, form a monoid, and that the subset of final states is a closed homomorphic image of the strings recognized by the machine. Reflecting the agenda of Bourbaki's algebra, he then moved from monoids to semi-rings by expressing the sequences recognized by an automaton as a formal power series in non-commutative variables. On analogy with power series in real variables, he distinguished among "rational" and "algebraic"

¹³⁹ Michael O. Rabin and Dana S. Scott, "Finite automata and their decision problems," *IBM Journal of Research and Development*, 3/2 (April, 1959), 114.

¹⁴⁰ Stephen C. Kleene, "Representation of Events in Nerve Nets and Finite Automata," in *Automata Studies*, ed. C.E. Shannon and J. McCarthy (Princeton, 1956), 37.

¹⁴¹ George A. Miller dates the beginnings of cognitive science from this symposium, which included a presentation by Allen Newell and Herbert Simon of their "Logic Theory Machine," Noam Chomsky's "Three Models for the Description of Language," and Miller's own "Human Memory and the Storage of Information." Ironically for Chomsky and Miller, the symposium marked a turn away from information theory as the model for their work.

power series and explored various families of the latter.¹⁴² Initially he identified “rational” power series with finite automata and hence finite-state languages, but could point to no language or machine corresponding to the “algebraic” series. That would come through his collaboration with linguist Noam Chomsky. [More on automata]¹⁴³

Formal Languages

Quite independently of these developments, Chomsky was pursuing his vision of a new mathematical linguistics based on syntactical structures. He introduced his ideas in 1956 in “Three Models of Language:”

*The grammar of a language can be viewed as a theory of the structure of this language. Any scientific theory is based on a certain finite set of observations and, by establishing general laws stated in terms of certain hypothetical constructs, it attempts to account for these observations, to show how they are interrelated, and to predict an indefinite number of new phenomena. A mathematical theory has the additional property that predictions follow rigorously from the body of theory.*¹⁴⁴

Here the three models were finite Markov processes (or finite-state languages), phrase-structure grammars, and transformational grammars. Following work on finite-state languages with psychologist George Miller, Chomsky articulated the phrase-structure grammars in greater detail in 1959 in his classic paper, “Certain Formal Properties of Grammars.” There he introduced the now central “Chomsky hierarchy” of finite-state, context-free, context-sensitive, and recursively enumerable (Turing) languages. At the time, only the first and the last had machine models. Indeed, he noted, “[I]nterest in structural properties of natural language thus serves as an empirical motivation for investigation of devices with more generative power than finite automata, and more special structure than Turing machines.”¹⁴⁵

Although Chomsky himself had concluded that phrase-structure grammars were too limited to account for natural languages and had moved on to transformational grammars, the phrase structure hierarchy immediately caught the attention of the computing community. The context-free grammars in particular struck resonant chords with work on automatic syntax analysis for machine translation and for compilation of high-level programming languages. Indeed, it quickly turned out that one of those “devices with more generative power” was already in widespread use, but not yet recognized as such. It was the stack, or last-in-first-out (LIFO) store first devised by Friedrich Bauer and Klaus Samelson, and was rapidly adopted by programmers as a means of holding intermediate results in the parsing of sequential formulas and in the “predictive analysis” of the syntax of natural languages in machine translation.¹⁴⁶

¹⁴² M.P. Schützenberger, “Un Problème de la Théorie des Automates,” *Séminaire Dubreil-Pisot* 13 (1959/60), No. 3, (23 November 1959); “On the Definition of a Family of Automata,” *Information and Control* 4 (1961): 245-70; “Certain Elementary Families of Automata,” in *Mathematical Theory of Automata*, ed. Jerome Fox (Brooklyn, NY 1963), 139.

¹⁴³ Link to the section on the theory of automata in my “Computer Science: The Search for a Mathematical Theory,” in *Science in the 20th Century*, ed. John Krige and Dominique Pestre (Amsterdam, 1997), Chap. 31. [http://www.princeton.edu/~mike/articles/20thcSci/20thcent.html#cra_automata]

¹⁴⁴ Noam Chomsky, “Three Models of Language,” *IRE Transactions in Information Theory* 2/3 (1956): 113.

¹⁴⁵ Noam Chomsky, “On Certain Formal Properties of Grammars,” *Information and Control* 2 (1959): 139.

¹⁴⁶ K. Samelson and F.L. Bauer, “Sequential Formula Translation,” *Communications of the ACM* 3/2 (1960): 76-83; F.L. Bauer, “The Cellar Principle of State Transition and Storage Allocation,” *Annals of the History of Computing* 12/1 (1990): 41-49.

Anthony Oettinger took steps to formalize the concept in 1960 with his “Automatic Syntax Analysis and the Pushdown Store,” and Chomsky followed a year later with a report on “Context-Free Grammars and Pushdown Storage.”¹⁴⁷ At about the same time Schützenberger showed that context-free languages corresponded to his “algebraic” power series, thereby confirming by his algebraic techniques what Seymour Ginsburg and H.G. Rice had shown about context-free portions of Algol by treating their expression in Backus normal form (BNF) as systems of set-theoretic equations.¹⁴⁸ Chomsky and Schützenberger then brought their results together in their classic paper on the subject.¹⁴⁹

In discussing the pushdown automaton, Chomsky noted that it was a special case of a linear-bounded automaton, as studied by John Myhill in 1960. This was the class of automata corresponding to computations for which an upper bound on the amount of tape was required. In 1963, Peter S. Landweber made the first connection between such automata and context-sensitive languages, and in 1964, S.Y. Kuroda established that full correspondence between the languages and non-deterministic linear-bounded automata.¹⁵⁰ The algebraic structure of the languages has proved more elusive. [More on formal languages]¹⁵¹

The Mathematics of Computation

In looking toward a “mathematical-logical theory of automata,” von Neumann knew only that “it will have to be, from the mathematical point of view, comminatory rather than analytical.” Insofar as the theory that emerged was mathematical, it was algebraic. As J. Richard Büchi put it in 1966:

*If the definition of “finite automaton” is appropriately chosen, it turns out that all the basic concepts and results concerning structure and behavior of finite automata are in fact just special cases of the fundamental concepts (homomorphism, congruence relation, free algebra) and facts of abstract algebra. Automata theory is simply the theory of universal algebras (in the sense of Birkhoff) with unary operations, and with emphasis on finite algebras.*¹⁵²

¹⁴⁷ Oettinger, “Automatic Syntax Analysis and the Pushdown Store,” *Proceedings of Symposia in Applied Mathematics*, vol. 12 (Providence, 1961): 104-129; Chomsky, “Context-free Grammars and Pushdown Storage,” Research Laboratory in Electronics [MIT], *Quarterly Progress Report* 65 (15 April 62).

¹⁴⁸ “Two Families of Languages Related to ALGOL,” *Journal of the ACM* 9 (1962): 350-71. Robert Floyd showed that the language as a whole was not context-free in “On the Non-Existence of a Phrase Structure Grammar for ALGOL 60,” *Communications of the ACM* 5/9 (1962): 483-84.

¹⁴⁹ Noam Chomsky and M.P. Schützenberger, “The algebraic theory of context-free languages,” in *Computer Programming and Formal Systems*, ed. P. Braffort and D. Hirschberg (Amsterdam, 1963).

¹⁵⁰ Landweber, “Three Theorems on Phrase Structure Grammars of Type 1,” *Information and Control* 6 (1963): 131-36; Kuroda, “Classes of Languages and Linear Bounded Automata,” *Information and Control* 7 (1964), 207-23.

¹⁵¹ Link to the section on formal languages in my “Computer Science: The Search for a Mathematical Theory,” in *Science in the 20th Century*, ed. John Krige and Dominique Pestre (Amsterdam, 1997), Chap. 31. [http://www.princeton.edu/~mike/articles/20thSci/20thcent.html#cra_formallanguages]; see also Sheila A. Greibach, “Formal Languages: Origins and Directions,” *Annals of the History of Computing* 3/1 (1981): 14-41.

¹⁵² J. Richard Büchi, “Algebraic Theory of Feedback in Discrete Systems, Part 1,” in *Automata Theory*, ed. E.R. Caianello (New York, 1966). The reference to [Garrett] Birkhoff is to his *Lattice Theory* (New York, 1948).

Indeed, the theory itself became part of what, by 1969, Garrett Birkhoff himself called “applied algebra.”¹⁵³ In historical perspective, it is striking that the most abstract concepts of modern algebra B semigroups, formal power series, lattices, categories B have provided the fundamental models for computation and, thus, for the most pervasive technology of our era. Schützenberger himself was surprised at first, noting in his paper on semigroups and codes that:

*It is particularly remarkable that the fundamental concepts of [the theory of semigroups], introduced by P[ierre] Dubreil in 1941, and studied by him and his group from an abstract point of view, should have immediate and important interpretations on the level of the concrete realization of coding machines and transducers.*¹⁵⁴

The traffic has not been simply one way. In the theory of computation, mathematicians found themselves in the unexpectedly complex realm of the finite but intractable large, a discrete realm which posed all the difficulties of the infinite with none of the compensating features of the continuous. It has provided unexpected and interesting challenges. [More on computers and mathematics]¹⁵⁵

Yet, looking at general texts currently in use, it is striking how little of that mathematics the computer science community has incorporated into the theory of automata and formal languages as it is presented to students. Instead, the field has tended to share the view of Michael Rabin, an algebraist who nonetheless, in the same volume, expressed his “personal bias that whatever generalization of the finite automaton concept we may want to consider, it ought to be based on some intuitive notion of machines. This is probably the best guide to fruitful generalizations and problems, and our intuition about behavior of machines will then be helpful in conjecturing and proving theorems.”¹⁵⁶ Again, it is a matter of agendas.

Michael S. Mahoney is Professor of History and History of Science at Princeton. He divides his research and teaching between the development of the mathematical sciences in the 17th century and the history of modern technology with a focus on the early development of theoretical computer science and software engineering. *Author’s address:* Program in History of Science, Princeton University, Princeton, NJ 08544 USA. Email: mike@princeton.edu.

¹⁵³ For example, Garrett Birkhoff and Thomas C. Bartee, *Modern Applied Algebra* (New York, 1970); Rudolf Lidl and Gunter Pilz, *Applied Abstract Algebra* (New York, 1984).

¹⁵⁴ M.P. Schützenberger, “Une Théorie Algébrique du Codage,” *Séminaire, P. Dubreil et C. Pisot* (Faculté des Sciences de Paris), Année 1955/56, No. 15 (dated 27 February 1956), p. 15-02.

¹⁵⁵ Link to portion of Michael S. Mahoney, “The Structures of Computation,” in *The First Computers—History and Architectures*, ed. Raul Rojas and Ulf Hashagen (Cambridge, MA, 2000). [http://www.princeton.edu/~mike/articles/ichc/ichc.htm#cra_compmath]

¹⁵⁶ Michael O. Rabin, “Lectures on Classical and Probabilistic Automata,” in Caianello, 306.

22. History in the Study of Computer Architecture

John Impagliazzo
Hofstra University

Abstract

The purpose of this work seeks to encourage alternate ways to present a course in computer architecture. Intended for post-secondary teachers of computing subjects, it attempts to show ways in which computing history can bring excitement and stimulation of knowledge by connecting technology with people, places, and events. The work also attempts to identify the landscape of computer architecture and suggests applications for today and for the future. This backdrop of highly technical topics can overwhelm students and make courses in the subject a daunting experience for them. Hence, this work suggests how teachers can use history to make topics in computer architecture more exciting and help spark a greater interest from their students.

Teaching technical courses at the college or university level can be challenging. With any course, presentations in a usual and mundane fashion discourage inquiry and student interest. Instructors often seek ways in which the course presentation could be more meaningful and interesting to students, such as including 'hands-on' activities, engaging in Socratic dialogues, and demonstrating videos on the subject. These and other methods are all useful and instructors should explore them further.

Another way to make technical courses more interesting is to use history. For computing courses, the use of computing history can be a natural method to use for such presentations because it connects technical material with people, places, and events. This is especially true for courses presented in a liberal arts setting. Indeed, it is most important for such courses in a technological or engineering setting.

Computing history places the topic or subject in a social context. Students can relate to human events and can easily appreciate the dynamics and reasons that influence a technical subject. When teachers surround a technical topic with a story or some historical event, students listen and develop a greater appreciation and understanding of it. In computer architecture, such contexts are natural and teachers should exploit the possibilities. Hence, we direct this work toward post-secondary teachers of computer architecture or related courses with the purpose of adding a new dimension of learning for their students.

The Subject of Computer Architecture

So, what is computer architecture? The answer to such a question is quite complex. The phrase "computer architecture" first appeared in 1964 when IBM developed its 360 machine. Fred Brooks used it to represent the instruction set used directly by a programmer. According to Brooks and others, it represented ".the structure of a computer that a machine language programmer must understand to write a correct (timing independent) program for that machine."¹⁵⁷ The implication of this statement is that someone programming in machine language needed to be conscious of the compatibility of the program with the machine. It also

¹⁵⁷ G.M. Amdahl, A. Blaauw, and F.P. Brooks, "Architecture of the IBM System 360," *IBM Journal of Research and Development* 8/2 (April 1964): 87-101.

implied that the time-independent nature of the machine allowed for different implementations. While this concept was not new at the time, it set the framework for how designers (architects) developed machines for future designs. For high-level-language computer architecture, it reduced the gap between languages and hardware. Although computer professionals did not universally endorse this concept even into the early 1970s, it did set the framework for new trends. Indeed, it led to new ideas regarding reduced instruction set computers, multiprocessing, and parallel processing architectures.

When we think of computer architecture, we naturally think of digital circuits, organization, assembly language, and data paths. We need to coordinate these elements with instruction sets and memory. Instruction sets involve arithmetic-logic instructions, memory instructions, and control (or transfer) instructions. With all these instructions, we need to design instruction formats and contrast the difference between physical memory and virtual memory schemes.

Memory lends itself to registers, primary or core memory, caching, and secondary memory. Integrating instructions with memory leads to instruction cycles with the typical fetch, decode, execute, and store scenarios. Optimization soon comes into play that leads to new designs, as in pipelining coupled with its performance metrics and its various types of execution hazards (conflicting data assignments) it can generate.

The quest for improved machine performance suggests areas such as parallel processing and networks. For pipeline architectures (used on most machines today), new breakthroughs occurred. These include branch prediction schemes, superscalar strategies, out-of-order executions, and synchronization. Pushing the envelope, we see new methodologies for caching, simulation methods, interrupt strategies, and the development of new architectures such as very large instruction word architectures.

Some Reactions and Actions

So, what would be the students' reaction to all of these topics? Unless they were highly motivated about the subject, the complexity of the topics could easily overwhelm them and they might find the subject matter non-stimulating and perhaps even boring. Typically, instructors would have students do the "normal" class activities of solving homework problems, taking quizzes, and doing examinations. In many cases, such activities further disengage students and classes become another hurdle to cross for the completion of a program or a degree. Many students often become unappreciative of the course (and perhaps the instructor), not because the subject or instructor was poor, but because its content and presentation was not interesting.

Instructors sometimes address this situation by exploring novel methods for teaching and learning or by varying the style of their presentations. The missing element may be the humanization of such a course. Instructors should relate the topics under question to human events, people, and places. They need to use computing history in such a course to make the material more accessible and relate it to relevant happenings. The people involved are the students, the teachers, and the pioneers of computing. The technological events can associate with social, political, and economic issues. Such elements of learning connect the real world with the curriculum at hand.

For courses in computer architecture, history seems to be a perfect fit. One way to make computer architecture more interesting to students is to use computing history as a theme transcending the course presentation. Teachers can introduce almost any topic in such courses

with a vignette, a story, or some current event. For example, a teacher might ask, “Who invented the computer?” The responses students might give in response are astonishing. Was it Konrad Zuse, John Atanasoff, John von Neumann, Charles Babbage, or some other person such as Bill Gates or Steve Jobs? It makes one wonder how many *teachers* of computing would know how to answer that question—an answer that is quite complex.

An interesting class activity might be to have students explore when and where certain topics first appeared. For example, a common input device used with personal computers is a mouse. Who invented it? When and where did that happen? What motivated its invention? Students are often motivated to find the answers to these questions and sometimes are startled to learn the results of their exploration. By engaging in this activity, students become more motivated and start to make connections with other technical events surrounding the computer field.

I wonder how many typical computing students have ever held, touched, or even seen a computer or its components. Here, we do not mean the casement of the machine. Do students know the difference between a resistor and a capacitor? Have they ever touched or seen a vacuum tube or transistor? Have computing students ever held a memory chip, a Pentium processor, or motherboard?

One would imagine that some such students have had the experience to answer affirmatively to at least one of these questions. Unfortunately, most students stereotype a computer as a box, with a monitor, keyboard, mouse, and other peripherals attached to it. Perhaps more teachers should do some “show-and-tell” activities in their computing classes so students are not ignorant of the machines they use. Teachers should have students disassemble components such as a diskette, a mouse, or a keyboard. They will marvel by what they learn. Students can then use computing history to explore how a mouse or keyboard came to be. They will also learn that most computers are not stereotypical and are present in a multitude of appliances and devices such as VCRs, televisions, automobiles, washing machines, cell phones, toasters, security systems, and a host of other possible examples.

Pioneering Ideas and Computers

To help instructors inject computing history in their architecture courses, let us explore a few developments that occurred over the years that may be of interest to learning. We have already mentioned memory, but its hierarchy, which is fundamental to computer architecture, is not a new idea. In fact, it was 1962 (a year before IBM specified its 360 computer) when Tom Kilburn first proposed the idea of memory hierarchy and demonstrated it on the Atlas computer at the University of Manchester in the UK.¹⁵⁸

¹⁵⁸T. Kilburn, D. Edwards, M. Lanigan, and F. Sumner, “One-level storage system,” *IRE Transactions on Electronic Computers* EC-11 (April 1962): 223-35.

In 1965, Maurice Wilkes proposed the idea of caching in his famous short paper. The method he proposed was what we know today as a direct-mapping cache.¹⁵⁹ It was at Cambridge University in the UK where we find the first implementation of a cache. In 1968, IBM announced the System 360 Model 85 computer, which was the first commercial machine that used caching to increase data transfer performance.¹⁶⁰ Both the Atlas and the IBM 360 used page protection in their operating systems. Virtual memory, however, provided an innovative and dynamic way to increase performance between primary and secondary memory. In 1972, IBM first implemented virtual memory on its 370 machine together with its VM operating system.¹⁶¹ Almost all computers today use virtual memory.

Addressing was another area with pioneering proportions. The EDSAC computer (1949), the UNIVAC I computer (1951), and the IAS computer (1952) all used accumulators that provided straightforward addressing. The Pegasus computer built by Ferranti in 1956 was the first to use general-purpose registers and brought addressing to new architectural levels.¹⁶²

In 1964, Control Data Corporation developed the CD 6600 machine, thought to be the first “supercomputer” of its time. This machine utilized multiple function units (units with the ability to do simultaneous operations), scoreboarding (a technique for instruction scheduling), and a time-shared pipeline.¹⁶³ These features demonstrated the ability of the machine to do dynamic scheduling (scheduling operations in real time), illustrating a novel methodology in pipelining. Real-time scheduling found its way in later computers. In 1967, Robert Tomasulo proposed an algorithm¹⁶⁴ that demonstrated another way to do dynamic scheduling. IBM implemented the algorithm on its 360/91 computer.

Following the 1960s, strategies and methods in pipelining continued to improve over the decades. Minimizing data hazards and structural hazards became prominent, as did strategies for optimizing branch hazards. Squeezing greater performance from a pipeline was and still is an important consideration. Superscaling (the ability to generate more than one instruction per clock cycle) became a challenge. In the mid-1980s, John Cocke of IBM demonstrated how to do this.¹⁶⁵ His efforts led to the IBM Power 1 processor and the continuing series of the IBM Power computers. The work of John Cocke also led to the development of reduced instruction set computers (RISC), a term now used to reflect this new architectural design.¹⁶⁶ Manufacturers have implemented the RISC concept in many of the computers in use today.

Superscaling became the mantra for issuing more than one instruction in parallel. The idea of multi-issue processors was not a new idea and the idea of using long instruction words

¹⁵⁹ M.V. Wilkes, “Slave Memories and Dynamic Storage Allocation,” *IEEE Transactions on Electronic Computers* EC-14/2 (April, 1965): 270-71.

¹⁶⁰ J.S. Liptay, “Structural Aspects of the System/360 Model 85, Part II: The Cache,” *IBM Systems Journal* 7/1 (1968): 15-21.

¹⁶¹ R. Case, and A. Padege, “The Architecture of the IBM System/370,” *Communications of the ACM* 21/1 (1978): 73-96.

¹⁶² J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, Third edition (San Francisco, 2003), 148.

¹⁶³ *Ibid.*, pp. A-68-69.

¹⁶⁴ D. Anderson, F. Sparacio, and R. Tomasulo, “The IBM 360 Model 91: Processor Philosophy and Instruction Handling,” *IBM Journal of Research and Development* 11/1 (January, 1967): 8-24.

¹⁶⁵ T. Agerwala and J. Cocke, “High Performance Reduced Instruction Set Processors,” *IBM Technical Report* (March, 1987).

¹⁶⁶ D. Patterson and D. Ditzel, “The Case for the Reduced Instruction Set Computer,” *Computer Architecture News* 8/6 (October, 1980): 25-33.

received much attention in the 1980s. One concept that began in the early 1980s was very long instruction word (VLIW) architectures. The advantage of VLIW is that it avoids certain data hazards that are eminent in pipelining. VLIW strategies place the burden of instruction issue on the compiler (software) rather than on the hardware. Some VLIW pioneers are Bob Rau who developed polycycle scheduling in 1982,¹⁶⁷ and Joseph Fisher who coined the VLIW idea in 1983.¹⁶⁸ In the mid-1980s, Rau developed the Cydrome VLIW computer. About the same time, John Ellis developed the Multiflow VLIW computer and showed its compiler intensity.¹⁶⁹ While neither of these machines achieved economic success, research continues at Intel and IBM to develop a true superscalar VLIW computer that one day may be economically competitive.

The Future from the Past

What can we expect in the future? It is not clear what the future has in store for computer architecture. The processors of today are quickly approaching their atomic limit. That is, as chips become smaller and more compact, the operation of these processors will be interfering with the molecular structure of its own composition. This should happen by 2015 to 2020. Hence, it will no longer be possible to make better machines with the materials currently in use. Chip manufacturers have known this for some time. The future lies in new concepts and ideas such as laser computers, optical switching, biological (DNA) computers, quantum computers, and other innovations.

All of this bodes well for computing and presents great opportunities for computing students and faculty members. To meet this awesome challenge, students will need more formalism in their studies and will have to engage in critical thinking and problem solving. They will need more mathematics and science to meet future challenges. Indeed, they will need to reflect introspectively on those challenges of the past that were successful and those that have failed. Students will need to engage in the evaluation of choices and be mindful of historical situations that influence the challenge at hand. Computing history may well be the greatest asset for students, scholars, inventors, and manufacturers. The inclusion of history in computer architecture is one way to broaden the breadth of computing knowledge and its implications for the future.

Conclusion

Teaching technical courses at the college or university level often requires novel and interesting ways to engage students in the subject. History can make technical courses, in particular computing courses, more interesting. Computing history is a natural way to present technical material associated with people, places, and events. This is especially true for computer architecture courses, where the topics are rich in historical development. Computing history places the subject in a social context and allows students to relate to human events. To

¹⁶⁷ B. Rau, C. Glaeser, and R. Picard, "Efficient Code Generation for Horizontal Architectures: Compiler Techniques and Architectural Support," *Proceedings of the Ninth Symposium on Computer Architecture* (April, 1982): 131-39.

¹⁶⁸ J.A. Fisher, "Very Long Instruction Word Architectures and ELI-512," *Proceedings of the Tenth Symposium on Computer Architecture* (June, 1983): 140-50.

¹⁶⁹ J.R. Ellis, *Bulldog: A Compiler for VLIW Architectures* (Cambridge, MA, 1986).

explore ways in which to use history in the computing curriculum, consult the IFIP report on the subject and published in the *Annals*.¹⁷⁰

When engaging in computing history activities, we must be mindful of two situations or problems. First, it soon becomes obvious that much of the development of computers occurred in the United States and in the United Kingdom. This can be problematic because the US/UK centrality, at least in the development of computing, can cast a negative social context in the view of other cultures and nations. In addition, this centrality may lead one to ignore the computing advances and developments made in other areas of the world. For example, how much attention do teachers pay to the development of computing in Russia and its development of the BESM machine? Doron Swade, senior curator of computing and information technology at the National Museum of Science and Industry in London, suggests the possibility that the Russian BESM-6 supercomputer may have been superior to Western computers during the Cold War.¹⁷¹ I wonder how many teachers thought of that prospect.

The second situation of caution is that computing history often focuses on hardware advances in the computing field. We need to understand that theory and software have also had inventions and advances. Perhaps because it is difficult to display tangible artifacts for these two areas of computing, they have not received their rightful place in the full development of computing history. Such sensitivity is important if teachers are to treat history in its proper way.

Being mindful of the above situations, the evolutionary development of current machines, and the daunting prospects of future computers, teachers have many options to make their classes in computer architecture interesting. The history of computing can provide an enriching theme throughout a computer architecture course to make the experience exciting. Students can relate the successes (and the failures) of the past to people, places, and events. These experiences can assist them in becoming an informed participant in the development of the computers of tomorrow. Would it not be grand if all computing courses had a history theme embedded within them?

John Impagliazzo (Ph.D.) is Professor of Computer Science at Hofstra University in New York. His study and work in computing spans four decades. He has authored or co-authored six books, has authored dozens of papers and has made over seventy professional presentations in computing and mathematics around the world, including a weeklong presentation at the International Centre for Theoretical Physics in Europe. His activity within ACM includes its accreditation committee that he has chaired for eleven years, the chair of the 1996 SIGCSE Technical Symposium, and the creation of the ACM Two-Year College Education Committee, which he chaired for three years. He is editor-in-chief of the publication *inroads*-the ACM SIGCSE Bulletin, a position he has held since 1997. Different universities, governmental agencies, and institutions around the world have called upon him as an expert in computing curriculum and accreditation. He is a longtime advocate of computing history. He currently chairs the IFIP Working Group on the History of Computing and a member and treasurer of the IEEE History Committee. *Author's address:* Dept. of Computer Science 103, Hofstra University, Hempstead, NY 11549-1030 USA. Email: cscjzi@Hofstra.edu.

¹⁷⁰ J. Impagliazzo, M. Campbell-Kelly, G. Davies, J.A.N. Lee, M. Williams, "History in the Computing Curriculum," IFIP (TC3-WG9.7 Task Group) Report, *IEEE Annals of the History of Computing*, 21/1 (January 1999): 1-15.

¹⁷¹ Doron Swade, "Back in the U.S.S.R.," *Inc. Magazine* (June, 1996). (<http://www.inc.com/magazine/19960615/1967.html>).

23. Bringing Women into the Curriculum

Jennifer S. Light
Northwestern University

Abstract

Computer scientists interested in using history to improve undergraduate education will make choices about which historical studies to include in their courses, and this paper offers encouragement that women's role in computing be one of the topics under consideration. By choosing examples from history that include women's participation, computer scientists can add an additional weapon to the recruitment and retention arsenal. Providing another set of role models, and opening the classroom to discussion about gender issues in the computing field, past and present, history can help to end rather than to reinforce the conventional stereotypes that women and computing do not generally mix. The following case study of the invention of the ENIAC offers instructors a sample fifteen-minute lecture that might be incorporated into a course on programming, computer architecture, or even a general introduction to the computing field.

Over the past two decades, the computer science community has expressed serious concern that the discipline has remained overwhelmingly male. The profession has undertaken a large-scale effort to support women's continued participation, developing resources ranging from committees on the status of women at the Computing Research Association and the Association of Computing Machinery to online communities for female computer scientists, such as SYSTERS and ResearchHers. Individual campuses, too, have tried to address this issue in a variety of ways by conducting studies on the shrinking pipeline, recruiting more female faculty, establishing mentoring programs for female undergraduates, and sensitizing male faculty and students to the subtle ways in which they might create a friendlier instructional climate.¹⁷²

Computer scientists interested in using history to improve undergraduate education will make choices about which historical studies to include in their courses, and this paper offers encouragement that women's role in computing be one of the topics under consideration. By choosing examples from history that include women's participation—Grace Hopper and Betty Holberton as part of the development of programming languages, for example, or the women of the Institute for Advanced Study—computer scientists can add an additional weapon to the recruitment and retention arsenal. Providing another set of role models, and opening the classroom to discussion about gender issues in the computing field, past and present, history can help to end rather than to reinforce the conventional stereotypes that women and computing generally do not mix. For as historians of computing increasingly have come to understand, despite their smaller numbers and lesser recognition, women already have played vital roles in the origins and evolution of the computing profession, in the United States and abroad.

The following case study of the invention of the ENIAC offers instructors a sample fifteen-minute lecture that might be incorporated into a course on programming, computer

¹⁷² Amy Pearl, et al. "Becoming a Computer Scientist: A Report by the ACM Committee on the Status of Women in Computing Science," *Communications of the ACM* 33/11 (1990): 47-58; Ellen Spertus, "Why Are There So Few Female Computer Scientists?" *MIT Artificial Intelligence Laboratory Technical Report* 1315 (Cambridge, MA, 1991); Tracy Camp, "The Incredible Shrinking Pipeline," *Communications of the ACM* 40/10 (1997): 103-10; Jane Margolis and Allan Fisher, *Unlocking the Clubhouse: Women in Computing* (Cambridge, MA, 2001). Several of these and related reports are available via hyperlinks from an online bibliography at: <http://cse.stanford.edu/class/cs201/projects-00-01/women-in-cs/links.html>

architecture, or even a general introduction to the computing field.¹⁷³ The case study proceeds with the assumption that presenting material on gender in the computer science curriculum can be a bit more delicate than other topics, and that the lecture is best framed not as “an introduction to gender and computing” (often a bore or turnoff for male students) but rather as a lecture about “the history of programming” or “the relationship between military funding and the direction of technological innovation.”¹⁷⁴ Discussion questions at the end offer instructors the opportunity to pick up on the gender issues in the story in the classroom discussion that follows.

Case Study: Women in Early Computing History

(Written in the voice of a faculty member lecturing to his or her class; readers should feel free to use it verbatim)

Good morning everyone. I'd like to begin our lecture today with a question for you, one I wonder if you've ever thought about. Does anyone know why we call computers “computers”? (Generally this is met with silence.) Today I want to devote some lecture time to answering this question. In doing so, I'll tell you about the origins of programming. It's a fascinating episode in computing history and we'll follow it up with some discussion about the extent to which the computing profession looks similar or different today, more than fifty years later.

Computers are machines developed to automate jobs formerly done by humans. If you asked someone what a computer was in 1944, they would have pointed to a person. If you asked them a few years later, at the end of the decade, they would have told you a computer was a machine; the people who formerly worked as human computers became redefined as computer programmers. It won't surprise anyone if I tell you that in today's world, computer programmers are more often than not men, and the occupation has something of a nerdy cachet. Yet programming was in its origins a feminized clerical occupation. So I want to tell you a little about how computing went from a human occupation to a task done by a machine, in this case the ENIAC, the first electronic computer in the United States. I'll also describe how programming first emerged as a female occupation.

So let me begin the story with a bit of historical background. Like the atomic bomb, the communication satellite, the Internet, and so many of the nation's scientific and technological innovations, the ENIAC was the offspring of an alliance between a university and the armed forces. In this case the partnership got underway during World War II between the Moore School of Electrical Engineering at the University of Pennsylvania and the US Army Proving Ground (APG) in Aberdeen, Maryland. At Aberdeen there was a Ballistic Research Laboratory dedicated to producing range tables for gunners. Now I don't expect you to know what a range table is so let me say a word about what this was.

In order to hit a target a gunner has to raise a weapon barrel to the proper angle. Many factors, such as air temperature and wind speed, go into the decision. Since gunners can't very well sit down on a battlefield and work out differential equations, the Army offered a pamphlet

¹⁷³ It is drawn from Jennifer Light, “When Computers Were Women,” *Technology and Culture*, 40 (1999): 455-83.

¹⁷⁴ Bracketing gender and race as lecture and discussion topics is certainly a common thing to do, but has its downsides. Students in the underrepresented group may feel singled out for their group's underachievement. Those in the majority may feel alienated from the topic or labeled as “oppressors.” Over the years this instructor has found such an approach makes it possible to tackle important issues in a way that engages a larger proportion of the class, while reducing some of the tension in discussion. Students repeatedly have reported in classroom evaluations and more informally that they appreciate this approach to the materials.

with firing tables to do the job. During the War the lab and the Moore school joined forces to recruit approximately two hundred women, many of them college graduates, to work as ballistics computers, hand-calculating these firing tables for rockets and artillery shells. The human computers used slide rules, calculators, and a sophisticated calculating machine called a differential analyzer, a precursor to the computer in its machine form. They worked at a high level of mathematical skill, for example, solving nonlinear differential equations in several variables. But in those days, their job—“computer”—was categorized as a clerical occupation.

Now we'll move on to the events leading to the invention of the computer in its machine form. Like so many later advances in the computing field, the first electronic computer was created to solve a specific problem, a point we often forget given the machine's multiplicity of applications in recent years. The problem that had to be solved was speeding the production of range tables. By 1943, over a hundred civilian women were working as human computers for the joint Aberdeen-Moore School operation. But that was still not enough. Although the Ballistics Research Laboratory recruited women from across the Northeast, the army needed firing tables faster than the human computers could supply them using available technologies. Construction for new weapons was being held up because it was physically impossible to supply manufacturers with the ballistic data. You can imagine during a wartime scenario what kind of major, urgent problem it was that weapons production was being slowed. So as a short-term solution, about a hundred members of the Women's Army Corps (known as the WACs) were assigned to begin training in Math for Ballistics Computations through the Moore School.

Yet even as the WAC courses went on, Moore School engineers were designing a longer-term solution to the computing crisis. This would be a machine to automate the production process for the same firing and bombing tables calculated by the human computers. This machine would become known as ENIAC, an acronym for the Electronic Numerical Integrator and Calculator (or Computer). ENIAC occupied a room that was 30 feet by 60 feet, weighed thirty tons, and consumed about the same amount of energy as a 174 horsepower motor.

By early 1945, engineers had assembled much of ENIAC's hardware. But the hardware couldn't do everything itself. Programming equations into the machine required human labor. The eventual transfer of computing from human to machine led to shifting job definitions. So a 'computer' was a human being until approximately 1945. After that date, the term referred to a machine. The former human computers were reclassified as programmers, also referred to as computer operators.

When we look at who was doing what on the ENIAC we see a fundamental distinction made between hardware and software. Designing hardware was a man's job. Software programming, considered to be a secondary, clerical task, did not match the importance of constructing the machine. As I've shared with you, computers—like office machine operators—worked in a feminized occupation. So in a sense it seems logical that in July 1945, six of the human computers would be reassigned to a similar but new occupation operating a gigantic computing machine. These first six programmers were Frances Bilas, Betty Jean Jennings, Ruth Lichterman, Elizabeth Snyder, Marlyn Wescoff, and Kathleen McNulty.

Yet if engineers originally conceived of the task of programming as merely clerical, it proved much more complex. When we look at what the six programmers had to do to learn to work the machine we discover they became intimately familiar with ENIAC's hardware, studying its circuitry, logic, physical structure and operation. They had to crawl around inside its massive frame to locate burnt-out vacuum tubes, shorted connections, and other bugs. And thus it was

the programmers who introduced the first users and problems to the machine. For example, physicists from Los Alamos came to Philadelphia to calculate the consequences of an H-bomb explosion. Their calculation required thousands of steps for the program and a million punch cards for the data. Since ENIAC could not store programs or remember more than twenty 10-digit numbers, the programmers printed out intermediate results before re-punching new cards and submitting them to the machine. Hardware designers, then, had done groundbreaking work, but it was the early programmers who made it possible to actually use the machine.

Now the ENIAC was designed and constructed in military secrecy. After the War, in February 1946, it was prepared for public unveiling. A press conference and formal dedication featured demonstrations of the machine's capabilities, created by the programmers. And so let me say a bit about how this invention was presented to the public. When Microsoft releases a new product, Bill Gates and his colleagues often put together a formal demo for the press so that he can get free publicity and a chance to educate people about the product. Well this kind of demo goes all the way back to the first electronic computer. Engineers staged a well-rehearsed event, cooperating with the US War Department (that's the precursor to today's Defense Department) which controlled representations of the project through press releases to radio and newspapers. The legacy we find in history books reflects this control. It focuses on hardware designers John W. Mauchly and J. Presper Eckert, Penn engineers, and Herman Goldstine, an Army mathematician—not on the early programmers who made the hardware work.

Newspapers parroted the press releases, and hailed hardware designers as having “fathered” the machine. They characterized ENIAC's ability to perform tasks as “intelligent,” but the human computers' work on the same computing tasks did not receive similar acclaim. So, for example, *The New York Times* of February 14, 1946 recounted details of the demonstration: “The ENIAC was then told to solve a difficult problem that would have required several weeks' work by a trained man. The ENIAC did it in exactly 15 seconds.” This “15 seconds” claim ignores the substantial time female programmers spent setting up each problem on the machine. Let me be specific here. It took between two and three days to set up an initial problem on the machine. Once an initial problem was set, with a few changes made to switches and cables the ENIAC could process another firing table soon after.

The next day's *Times* carried photographs of Eckert and Mauchly and a report that: “[The machine is] doing easily what had been done laboriously by many trained men...Had it not been available the job would have kept busy 100 trained men for a whole year.” While this article alluded to the participation of many individuals other than Eckert, Mauchly, and Goldstine, one can't help noticing that the hypothetical hundred are described as male. Why did neither article report that the machine easily did calculations that would have kept one hundred trained women busy? Now all of you in the room today know that historically the default term for people was the word ‘men.’ Nonetheless, the omission of women here is surprising, since the job of computer, like secretary, was widely regarded as women's work.

What about photographs? Well, the programmers vanished from these as well. One publicity photo accompanying *The New York Times* story foregrounds a man in uniform plugging wires into a machine. Notice how the figures of two programmers in the background can be seen only by close scrutiny. Thus, the press conference and follow-up coverage rendered invisible both the high-skill labor required to set up the demonstration, and the gender of the programmers who did it.

The role of the War Department and media in shaping public discourse about the machine is significant. Several potential opportunities for the women to get some public attention never materialized. For example, this publicity photo of the ENIAC was among the most widely disseminated images of the machine. When it was published as an Army recruitment ad, the women were cropped out. Now this action is understandable, at one level, since the ENIAC programmers were all civilians. Yet given the important participation of the Women's Army Corps in closely related wartime work, this constituted another missed opportunity.

Discussion Questions

Having answered the question "Why do we call computers computers?" and shared with students the early history of programming, the instructor may wish to conclude the lesson at this point. Alternatively, he or she may wish to open the floor to discussion. Below are three sets of discussion questions to stimulate students to link the historical study of the ENIAC to contemporary issues in the computing field, including but not limited to gender. Answers are suggestive, rather than definitive, and include pointers for further reading.

Question: The ENIAC project made a fundamental distinction between hardware and software. Designing hardware was a man's job. Software programming, a secondary, clerical task, did not match the importance of constructing the machine. Are there similar divisions of labor in the computing field today? Do you think the field of computer science would be substantially different today had it been publicly recognized earlier that the first programmers were female? How?

Answer: Certainly there are gendered divisions of labor in the computing field today. One place to go for the latest statistics is the NSF's official listing of its research reports on the status of women, minorities, and people with disabilities in science and engineering (URL: <http://www.nsf.gov/sbe/srs/women/start.htm>). A less-up-to-date but interesting account of the masculinization and feminization of specific computing subfields can be found in Rosemary Wright and Jerry A. Jacobs, "Male Flight from Computer Work: A New Look at Occupational Resegregation and Ghettoization," *American Sociological Review* 59 (1994): 511-36. The final question is obviously a more speculative discussion; certainly it is an opportunity to raise awareness that women have been part of the computing field from its origins.

Question: What is the role of the military, and war, in stimulating technological innovation? To what extent have military needs and military culture influenced the nature of computing technologies?

Answer: Military and government needs have played central roles in the history of computing, and raising this question offers the opportunity to acquaint students with the larger cultures of government and private funding that have influenced and continue to influence the direction of computing research, such that they recognize how larger forces—oftentimes the solution to a particular technical problem—rather than the most interesting technical questions shape the trajectory of computing history. Paul Edwards' *The Closed World* (Cambridge, 1996) is an excellent place to start for the history of computing in general and artificial intelligence in particular. Janet Abbate's *Inventing the Internet* (Cambridge, 1999) can provide a history of the Internet in this larger context.

Question: To what extent do public demos heralding the launch of a new computer product exert the same effect as the ENIAC demo, shaping public knowledge about the innovation?

Answer: In many cases, a great deal. Raising this question gives students the chance to reflect on the fact that the Bill Gates and Thomas Edisons of the world are few and far between, and get a great deal of credit for large-scale projects, and that they, as computing professionals, are more likely to be involved in team-level research. Who in fact gets the credit for scientific and technological innovation is often linked to gender, as Margaret Rossiter documents in “The Matthew Matilda Effect in Science,” *Social Studies of Science* 23 (1993): 325-41.

Conclusion

The case study and discussion questions above provide a model of the kind of historical case study that can accomplish multiple curricular goals: introducing students to key issues in the computing field, past and present, and simultaneously encouraging them to recognize how women have been important participants in computer science throughout its history. Women’s role in the development of ENIAC is not the only story of pioneering female participation in the computing field, and for instructors seeking additional resources, Janet Abbate’s introductory essay to a recent special issue of the *IEEE Annals of the History of Computing* (October/December 2003)¹⁷⁵ on women in computing is the best place to get started, with its concise overview of existing research and pointers to articles on women’s work in the development and applications of hardware and software, in the United States and abroad, and in academic and corporate settings.

Jennifer S. Light is an assistant professor at the School of Communication and the Departments of History and Sociology at Northwestern University. She is the author of *From Warfare to Welfare: Defense Intellectuals and Urban Problems in Cold War America* (Baltimore, 2003) and is currently working on a book on the history of information management in American cities. *Author’s address:* Northwestern University, School of Communication, 2240 Campus Drive, Evanston, IL 60208 USA. Email: light@northwestern.edu.

¹⁷⁵ Available online at: <http://csdl.computer.org/comp/mags/an/2003/04/a4004.pdf>

24. Human Computers and their Electronic Counterparts

David Alan Grier
George Washington University

Abstract

Before we had electronic computers, difficult scientific computations were handled by human computers. These computers had little direct influence on the development of the modern computer, yet they dealt with a number of issues that were also found in the development of computing machines. By studying human computers, we get a new perspective on the basic themes behind the electronic computer—the themes of labor division, process control, mass production, and professionalization.

In studying either the practice or the history of computing, few individuals venture into the landscape that existed before the pioneering machines of the 1940s: George Stibitz's complex calculator, John Attansoff's linear algebra machine, Konrad Zuse's Z1, Howard Aiken's Mark I. Arguably, there is a good reason for making a clean break with the prior era, as these computing machines seem to have little in common with scientific computation as it was practiced prior to 1940. At that time, the term "computer" referred to a job title, not to a machine. Large-scale computations were handled by offices of human computers.

Very few human computers made the transition to the age of electronic computers. In the mid-1940s, George Stibitz remarked that "Human agents will [soon] be referred to as 'operators' to distinguish them from 'computers' (Machines)."¹⁷⁶ The only large cohort of computers who became computer programmers was a group of slightly less than a dozen women at the Moore School of Electrical Engineering at the University of Pennsylvania. These women were initially trained on the school's Differential Analyzer and then moved to the ENIAC. They were far from typical computers, as they were trained to use a large computing machine from the start. Most human computers worked only with an adding machine or a logarithm table or a slide rule. During the 1940s and 1950s, when electronic computers were becoming more common in scientific establishments, human computers tended to view themselves as more closely connected to the scientists for whom they worked rather than to the engineers who were building the new machines and, hence, did not learn programming.

In spite of the break between human computers and their electronic counterparts, these workers illustrate many of the basic themes of the modern computer era and help us to understand the practice of computer science. In all, there are four broad ideas that can be explained by the stories of human computers. The first is the division of mental labor—the analysis and separation of individual tasks so they can be handled sequentially, like a conventional program, or in parallel. Parallel processing proves to be an early development in the history of human computing, and it points to the second idea, the control of parallel processes, that can be illustrated by this era. The great growth of human computing offices began as scientists learned how to divide computing tasks into small units, handle those units in parallel, and reassemble the results.

The third theme is that of standardization. Anyone who has taught the elementary sequence in computer science has encountered, at least once, a student who resisted using

¹⁷⁶ Paul Ceruzzi, *Reckoners: The Prehistory of the Digital Computer, From Relays to the Stored Program Concept, 1935-1945* (Westport, CT, 1983).

standard constructs, data structures, or algorithms. The story of human computation shows how the computers developed standard ideas as a means of saving labor—labor that could be long and arduous. The final theme of human computers is that of professionalization, which is the process of defining an independent body of knowledge, practices, standards, and organizations. The nature of computing as a professional occupation is currently obscure because we live at an odd juncture in its history where the idea that computer science might be a part of mathematics or electrical engineering is long forgotten. The traditional role of professional organizations as promoters of computer science is slipping away; information about computing is now more commonly dispensed by commercial presses and Internet chat groups than by professional societies. Returning to the work of human computers illustrates the need for social organizations to support a scientific endeavor.

In the end, the story of human computers is neither a quaint look at a bygone world nor a cautionary tale about how difficult computation can be. It illustrates the major themes of computation, and puts a human face on those themes.

The Division of Labor

“The greatest improvements in the productive powers of labour,” wrote the Scottish philosopher Adam Smith in 1776, “seem to have been the effects of the division of labour.” The division of labor occupied a central place in Smith’s treatise on economics, *The Wealth of Nations*. Smith argued that the division of labor was a key step for any society to take. It increased wealth by allowing individuals to concentrate on the tasks that he or she did best. In practice, this specialization would decrease the costs of production and allow people to produce more goods. Smith acknowledged that divided labor had a long history in Europe and that the basic concepts of divided labor might be applied to scientific research. With divided labor, “[e]ach individual becomes more expert in his own peculiar branch [of science],” he wrote, and “the quantity of science is considerably increased by it.”

During the second half of the eighteenth century, the time when Smith was writing *Wealth of Nations*, astronomers made the first attempts to divide the labor of scientific computation into smaller pieces and to find workers who might handle each piece. For the most part, these early computers did calculations that could stand alone as an independent computation, even though they were part of bigger problems. In the summer of 1757, three French astronomers divided the computation of the return date for Halley’s Comet among themselves. The problem was difficult to compute because it was a three body problem. To properly compute the orbit of the comet, the astronomers had to compute the relative positions of the Sun, Jupiter and Saturn, as these three bodies influenced the motion of the comet. In this calculation, two of the computers shared the work of computing the orbits of Jupiter and Saturn. The third handled the orbit of the comet. It was, according to one historian of the work “the first large-scale numerical integration ever performed.” It required about three months and produced an answer that was only thirty-one days from the actual date of return.¹⁷⁷

The first employers of human computers were the almanac offices that produced star charts for navigators and surveyors. In France, the almanac was produced by the Academie des Sciences and was called the *Connaissance des Temps*. It employed two of the astronomers who had worked on the computation for Halley’s Comet. An English almanac was created in 1767. This office employed five human computers. Four computed the positions of the planets

¹⁷⁷ Curtis Wilson, “Appendix: Clairaut’s Calculation of the Comet’s Return,” in *The General History of Astronomy*, ed. Rene Taton and Curtis Wilson (Cambridge, 1995), 83-86.

and the moon. Each set of calculations was done twice by independent workers. The fifth computer, called the comparator, looked for errors by comparing the duplicate answers.

The great advancement in divided mathematical labor came during the French Revolution in 1789, when the National Assembly developed the metric system of measurement. On the recommendation of the Academie des Sciences, the assembly considered mandating a decimal system on all forms of measurement, including the measurement of angles. Under this rule, a quarter-circle would be divided into 100 grads instead of 90 degrees. This idea created a substantial mathematical problem. The people who used trigonometry in their daily work, such as surveyors and navigators, employed tables of trigonometric functions in order to calculate the size of a piece of land or identify a location on the earth. Unless the government created a new set of trigonometry functions, the surveyors and navigators would have no incentive to use the new system. Without these tables, they would be forced to transform their measurements into degrees, use conventional trigonometry tables, and possibly transform some of the answers back into grads.

The task of preparing the new trigonometry tables was given to a civil engineer, Gaspard de Prony. De Prony was familiar with *Wealth of Nations* and Smith's discussion of divided labor. In his book, Smith had given an example of pin manufacturing to show how tasks could be divided into component parts. De Prony recognized that the computation of trigonometry tables could be divided into a series of additions through the use of calculus. He assembled a small staff drawn, in part, from the *Connaissance des Temps*, to analyze the computations for sines, cosines, and the other functions, and to create worksheets that would lead other workers through the additions that would create trigonometry tables. He then hired about ninety workers to prepare the tables by completing the worksheets. This group finished eighteen manuscript volumes of tables in a period of about three years.

In 1795, just as de Prony's computations were coming to their conclusion, the National Assembly decided that they would not require decimal angle measure as part of the new metric system. Once they made that decision, the new tables had limited value. De Prony made several attempts to have them published, but was never successful. They might have been completely forgotten had they not come to the attention of Charles Babbage in the 1820s. At that time, Babbage was a member of the Royal Astronomical Society in London and was engaged in the calculation of astronomical tables. He would create a computing plan for these tables and, like the Nautical Almanac Office, hire two independent computers to do the computation.¹⁷⁸

Babbage found that, even with two computers, his final calculations had more errors than he could tolerate, so he decided that he would build a geared machine to do the calculations. He approached this problem using the ideas of de Prony as his guide. He chose to compute the tables by the mathematical method of interpolation, the calculation of values between two known values. He divided the method of interpolation into its fundamental additions and designed a machine that could perform those additions. This machine was called the Difference Engine. In promoting this machine, he compared it directly to the accomplishments of de Prony, writing that de Prony had shown how "the division of labor can be applied with equal success to mental as well as to mechanical operations." He claimed that his Difference Engine could replace most of the ninety human computers that de Prony had employed.

¹⁷⁸ Computers used the term "plan" rather than "program" to describe their instructions for calculation. The term "program," in its modern sense, was introduced by the staff who worked on the ENIAC computer.

Babbage is almost universally acknowledged as the first person to have conceived of the programmable computer, but if you trace his influence on those who created the first electronic computers in the 1930s and 1940s, you will find that the path is filled with problems. Babbage never completed his Difference Engine. His later, more sophisticated machine, the Analytical Engine, was unknown to most of the early electronic computer designers. Those who did know of the work drew few ideas from it. However, if you trace Babbage's influence through human computers and the division of labor, his importance is much more obvious. Though Babbage never completed his Difference Engine, others were able to build such a device and used it to good effect. The American Nautical Office tested a difference engine in 1857, though they never adopted it for production work. Almost seventy-five years later, the director of the British Nautical Almanac discovered a commercial accounting machine that could be used as a difference engine. He quickly purchased one for his office and used it to produce tables for his publication. The American Nautical Almanac Office purchased a similar machine in 1940 and used it both to produce an almanac and to create tables for the military during World War II. Both directors were familiar with Babbage's discussions on divided labor and the importance of divided labor in computation.

Babbage described de Prony's method of divided labor for computation in his book, *On the Economy of Machinery and Manufactures* (1835). This book was widely read by those who created and managed factories in the nineteenth century. More than one critic has noticed that it updated the ideas of Adam Smith for the early machine age. In spite of its wide circulation, de Prony's computing office was not often duplicated. In effect, there were few problems that required such a large computing staff and such a complete division of labor. Most calculations could be handled by a staff of eight to twelve computers and a judicious use of "aids to computation" such as logarithm tables or, after the 1890s, of adding machines. De Prony's computing office was recreated only once, in 1938, when the U.S. Government established the Mathematical Tables Project.

The Mathematical Tables Project was a solution to a social problem rather than a scientific one. It was created by a relief agency, the Works Project Administration, as a means of employing out-of-work clerks in New York City. At its height, it employed a computing staff of 450 individuals, most of whom knew little arithmetic beyond the basic rules of addition. The project created twenty-eight volumes of mathematical tables, numerous smaller tables, and literally thousands of special computations for various agencies during World War II. It operated until 1948. During its last years, it operated as a surrogate for the ENIAC electronic computers. Several scientists, including John von Neumann, used the group to estimate how quickly the ENIAC might handle their problems.

Managing Computers

Scientists learned to manage large staffs of human computers at the same time that they learned to divide mathematical labor and assign it to individual computers. These scientists had to develop methods that would control production, keep the workers on a deadline, and help identify mistakes in the final results. The first managerial tool that they developed was the worksheet, a piece of paper that described each step of the calculation and had blank spaces for the computers to fill with their work. Such worksheets were used by the first director of the Nautical Almanac, Nevil Maskelyne, in the 1770s. Maskelyne would prepare worksheets by hand, drawing a grid on one side of a sheet of paper and writing instructions down the margins. Occasionally, he would illustrate the calculation with a little diagram on the back of the paper.

Maskelyne would send these sheets to his computers through the mail. The computers would complete the forms at their leisure and return them to Maskelyne by the same method.

Through the early nineteenth century, most computers worked in their homes. The scientist who employed these computers would coordinate production either by sending instructions through the mail or by occasionally meeting with the computer. This managerial technique saved the expense of a separate computing facility, but it did not allow much discipline over the computers. Computers could easily delay calculations and refuse to respond to letters. One computer of the 1850s was especially frustrating to the director of the American Nautical Almanac. When the computer refused to answer any letters, the director wrote to a mutual friend, "Will you do me the kindness to explain to [the computer], who does not understand the subject that his relation to me as a public officer, he receiving compensation from the government under an appointment conferred by me with authority required that he should keep me informed of his condition or if he finds that inconvenient or impracticable, that he had better resign his appointment which it will give me pleasure to renew, when he is again ready for work."¹⁷⁹

Eventually, the American Nautical Almanac pulled all of its computers into a central computing room. This practice had been introduced by the British Almanac in 1833 and had been followed by the establishment of a central computing facility at the Greenwich Observatory a few years later. Through these computer rooms, scientists could enforce a more rigorous discipline on their workers, though the discipline of the nineteenth century was still fairly informal and relaxed. One computer of the American Almanac computers recalled that: "In theory there was an understanding that each assistant was "expected" to be in the office five hours a day."¹⁸⁰

Once scientists began building computing offices, their next step was to develop broad methods of computation so that the staff could be kept busy by working on different kinds of problems. Throughout most of the nineteenth century, almanac computers and observatory computers were considered different kinds of mathematical workers. Observatory computers generally spent their days reducing data, taking the positions of stars and planets as recorded by observers at telescopes, and converting them into absolute coordinates in the celestial sphere. This kind of work was considered fairly elementary and commanded less pay than the more sophisticated orbital calculations that were handled by almanac computers. Yet, almanac calculations tended to be seasonal and often did not require a full year's effort by a computer. By combining observatory computations and almanac calculations, an astronomer could keep a computer fully occupied throughout the year.

The U.S. Naval Observatory was one of the first institutions to require computers to handle both observatory data reduction and orbital calculations for an almanac. In the 1890s, the director of this combined computing facility, Simon Newcomb, suggested that this computing operation should be pushed one step further. He proposed that American scientists should join together and create an "Institute for the Exact Sciences." He argued that this institute should help scientists build mathematical models for both the physical and newly developing social sciences. The centerpiece of this organization would be a large computing room that would provide free calculating services to the world's scientists. Newcomb was the best known

¹⁷⁹ Charles Henry Davis to Thomas Sherwin, Principal, Boston High School, July 5, 1850. Papers of the Nautical Almanac Office, Records of the Naval Observatory, Library of Congress, Washington, DC.

¹⁸⁰ Simon Newcomb, *The Reminiscences of an Astronomer* (Boston, 1903), 74.

American scientist of his day and perhaps the only individual with a reputation sufficient to lead such an organization; but he soon found that there was little enthusiasm for his project.

Mass-Produced Computing Machines

Through the end of the nineteenth century, most human computers did all of their work by hand. Other than a paper and pen, the only tool that they would regularly use was a table of logarithms. Geared adding machines had first appeared nearly two hundred years before, but few of the early adding machines were employed by human computers. These machines were too expensive and too fragile to be used in regular calculation. Human computers began to acquire adding machines only after the appearance of mass-produced machines in the 1880s. The first machine to be widely used by computers was the Arithmometer of Thomas de Colmar.

De Colmar, an actuary, marketed his machine to insurance companies and commercial firms, but it soon found its way into observatories, almanac offices, and university laboratories. "With the Arithmometer at hand," wrote one scientist, "the work [of scientific calculation] becomes rather amusement than labour, especially to those at all fond of ingenious and beautiful mechanism."¹⁸¹ These machines allowed computers to handle problems that were previously considered too lengthy to undertake. The most influential of these methods was that of least squares. Astronomers used least squares to prepare highly accurate calculations of orbits, but the technique had applications far beyond astronomy. Least squares calculations introduced organized calculation into surveying and social statistics. Surveyors used the technique to adjust their work and minimize errors in their final presentations. Statisticians used the method to analyze data to find underlying factors that seemed to influence physical and social phenomena. The method of least squares was promoted by Myrick Doolittle, the head of the computing division of the U.S. Coast and Geodetic Survey. Doolittle developed a particularly efficient plan for doing least squares computations. Least squares moved into statistical practice through the efforts of George Snedecor, a mathematician at Iowa State College, and Henry A. Wallace, a newspaper publisher in Des Moines. The two wrote a well-circulated book on least squares calculations. Snedecor created a large staff of human computers at his college and trained them in the method of least squares.

Computing machines forced human computers to develop a new set of skills. Almost all of these machines were designed for commercial work, for dealing with monetary calculations. In order to do scientific calculations, with their decimal numbers, complex mathematics and long divisions, computers often had to follow fairly detailed instructions. Though some individuals felt that calculating machines should be designed expressly for scientific purposes, most computers felt that the commercial machines were the best computing tools, in spite of their drawbacks. "[Commercial machines] are the product, not of a single and, perhaps, not too experienced designer but of a group of experts," wrote L.J. Comrie, the director of the British Nautical Almanac in the 1920s. He further noted that, compared with special-purpose scientific machines, "spare parts and expert service are readily available" and that these machines had prices "that are economical as compared with the overhead costs of design and construction on a small scale."¹⁸²

¹⁸¹ W. Stanley Jevons, "Remarks on the Statistical Use of the Arithmometer," *Journal of the Statistical Society of London*, 41/4 (December 1878): 597-601.

¹⁸² L.J. Comrie, "Inverse Interpolation and Scientific Applications of the National Accounting Machine," *Journal of the Royal Statistical Society*, 3/2 (1936): 87-113.

Comrie became well known for his skill at computation and his ability to adopt commercial machines for scientific purposes. He was the almanac director who discovered the accounting machine that could operate as a difference engine. He also developed procedures for doing scientific calculations with IBM punched card tabulators. The first tabulators that he employed were little more than adding machines, yet he showed how they could do the full range of arithmetical operations and could be used for orbital calculations.

Computing As a Formal Discipline

Between 1925 and about 1945, L.J. Comrie was perhaps the world's leading expert in scientific computation. For many of those years, he served on the Mathematical Tables Committee of the British Association for the Advancement of Science. This committee created tables of mathematical functions and published reference works on computation. It also served as a coordinating body, an organization that collected information about calculation. In the United States, the National Academy of Science operated a similar group, called the Subcommittee on the Bibliography of Mathematical Tables and Other Aids to Computation. This organization was concerned with documenting the literature of computation, both the methods of calculation and mathematical tables that could be used to simplify hard problems. Through the late 1930s, there was no unified literature of numerical methods. Scientists published their computational ideas in the journals of astronomy, electrical engineering, optics, statistics, and other fields.

Beginning in about 1939, the Subcommittee on Mathematical Tables and Other Aids to Computations moved to identify computation as an independent discipline. As an independent discipline, it would be applicable to all forms of scientific practice. It would have a standard set of computational methods that could be used for problems in any form of scientific practice. It would have a single literature that could be used as a reference by anyone undertaking scientific computation. Lastly, it would have a recognized way of training new computers. The leader of this effort was R. C. Archibald, the chair of the Subcommittee. He was aided in his efforts by Comrie, though Comrie's contributions were limited by the fact that he was an ocean away from Archibald and that he was undertaking computations for Great Britain's effort in World War II.

The centerpiece of Archibald's efforts was a new journal named after his committee, *Mathematical Tables and Other Aids to Computation*. The first issue of the journal appeared in the winter of 1943. The first several issues were produced almost exclusively by Archibald. "R.C.A. greatly regrets the apparent necessity for numerous personal contributions in this issue, as well as in the second," he wrote in his introduction to the debut number. "It seems certain that elimination in this regard shall be noticeably operative in the third and later issues."¹⁸³ His prediction proved true, as the journal quickly attracted a large audience. By 1945, *Mathematical Tables and Other Aids to Computation* had a subscription list of 300 and had published twelve issues. Its efforts were aided by a number of the wartime computing groups, notably the Mathematics Tables Project in New York City. Though the Mathematical Tables Project had originally resembled the computing organization of Gaspard de Prony, it had acquired adding machines during the war, reduced its computing staff, and operated as the general computing office of the American Office of Scientific Research and Development. It taught classes on computation, sponsored seminars on new computing methods, and circulated a mimeographed textbook on computation.

¹⁸³ R.C. Archibald, "Introductory," *Mathematical Tables and Other Aids to Computation*, 1/1 (1943): p. i.

The efforts to establish computing as practiced by human computers as an independent discipline reached its peak at the end of World War II. The U.S. Navy and U.S. Army provided funds to operate the Mathematical Tables Project as an independent computing office. The subcommittee on Mathematical Tables and Other Aids to Computation held a conference on computation at the Massachusetts Institute of Technology that assembled most of the leading wartime computers. However, to those computers who were paying attention to the postwar plans, the future of the human computer was far from certain. First, the demand for scientific computation was falling rapidly. War industries and agencies had employed several thousand computers between 1940 and 1945, but they had little use for these employees after the end of the war and released them.

Second, the new electronic computer promised to replace the human computers. By the summer of 1946, it was clear that human computers would have only a limited role in the post-war world. These workers were not invited to the conferences on electronic computers. Only a few found positions as programmers. The knowledge that they accumulated in the journal *Mathematical Tables and Other Aids to Computation* became part of the literature of numerical analysis. The journal itself was renamed *Mathematics of Computation*. The last major effort of the human computers was a publication called *Handbook of Mathematical Functions*, a book that described how to evaluate complicated mathematical expressions. It was published in the mid-1960s, a time when the computer industry was growing rapidly. Even though half of the pages were filled with the old mathematical tables, it became one of the best-selling scientific books of all time.¹⁸⁴

Conclusion: The Legacy of Human Computers

By the early 1950s, most of the wartime computers had vanished. Commonly, they took positions as clerks, salespeople, bookkeepers, teachers, or mothers after leaving their computing jobs. The programmers who took the place of human computers did different tasks, yet they rediscovered some of the most fundamental lessons of hand computers. Programming taught its practitioners how to divide large activities into small steps and then write the code to handle each step.

The other lessons of the human computer were repeated both in the social organizations around electronic computers and in the computers themselves. The computing centers of the first decades of the computing era bore at least a superficial resemblance to the computing floors of the Nautical Almanac Offices, Observatories, and the Mathematical Tables Project. The early programmers also learned the lessons of L.J. Comrie-lessons that taught how to use a business machine to do scientific work, how to make a small computer behave like a big computer, and how to make any machine do more than it was intended to do. Finally, the issues of professional organizations appeared shortly after the University of Pennsylvania announced the ENIAC computer. By early 1948, the first computer scientists were asking how they could disseminate information about computing machines, how they could train new programmers, and how they could control entry to this new field.

Those stories of electronic computers that appeared in the press of the late 1940s and early 1950s spoke of electronic brains, miracle machines, and unrealistic expectations of what these devices could do. By the time the public began to get a clearer understanding of how computers operated, they had generally forgotten the contributions of human computers; yet,

¹⁸⁴ See, David Alan Grier, "Irene Stegun, the Handbook of Mathematical Functions and the Lingering Influence of the New Deal," submitted to the *American Mathematical Monthly*.

the problems faced by the early computer scientists had parallels in the issues encountered by human computers in the seventeenth, eighteenth, and nineteenth centuries.

Selected Bibliography

This paper is based upon the author's book, *When Computers were Human* (Princeton, NJ: 2005), to be published by Princeton University Press. A considerably fuller description of all computing groups mentioned in this article can be found there. Babbage's *On Machinery and Manufacture* is available on the Internet from Project Gutenberg.

Examples of Computing Plans can be found in the published tables of the Mathematical Tables Project. The Mathematical Tables Project published twenty-eight volumes of tables, and at least one can usually be found in any college library. Sadly, they are not catalogued in a standard manner, but are often found listed under the authorship of Arnold Lowan, the Works Project Administration, or the Mathematical Tables Project. A complete list of these tables can be found at Grier, David, "The Math Tables Project: The Reluctant Start of the Computing Era," *IEEE Annals of the History of Computing*, vol. 20, no. 3, pp. 33-50.

The book, *A Handbook of Mathematical Functions* (Abramowitz and Stegun, Washington, D.C., National Bureau of Standards, 1964) was written by former human computers and suggests the kind of work that they did. The journal of human computers, *Mathematical Tables and Other Aids to Computation*, is available electronically through JSTOR. It also contains many interesting reports of early electronic computers.

L.J. Comrie has yet to be given the biography that he deserves. The best exposition of his life can be found in Croarken, Mary, *Early Scientific Computing in Britain*, Oxford University Press, 1992.

David Alan Grier is an associate professor at George Washington University and is the grandson of a human computer. (His grandmother, Blanch O'Kane Gallup, was one of "Glover's girls" at the University of Michigan and took a BA in mathematics in 1921.) He has just completed the book, *When Computers were Human* (Princeton, New Jersey, 2005) and is now studying the history of mathematical modeling in public policy. He is currently the editor-in-chief of the *IEEE Annals of the History of Computing*. *Author's address:* George Washington University, Washington, DC 20052 USA. Email: grier@gwu.edu.

25. Integrating Ethics into a Computing Curriculum: A Case Study of the Therac-25

Chuck Huff and Richard Brown
St. Olaf College

Abstract

Almost all computer ethics courses use cases to some extent. We show how we have integrated detailed historical cases into ethical reflection throughout the computer science major, and in particular in the course Ethical Issues in Software Design. We then present a particular case, that of a radiation therapy machine in the mid 1980s that killed several people with massive overdoses of radiation. The overdoses were caused by poor software design. To understand the failings of the software design process, we present significant technical and social detail of this case and show how this detail helps to make the case effective in teaching ethical issues.

Enough computer ethics courses have now been established in the field to have a distinctive content and several attempts at a recommended curriculum,¹⁸⁵ and to have generated several distinct approaches to class organization and practice. Almost all courses use cases to some extent in their approach, but some are more intentionally case-based and others are driven more by other pedagogical approaches. Some texts use a “controversies-in-the-field” approach,¹⁸⁶ with the primary aim of exposing students to the arguments on the different sides of the controversy. The most common approach¹⁸⁷ is topic-based, covering the standard topics such as privacy, liability, and intellectual property that appear in the recommended curricula. A minority approach is one based on social issues,¹⁸⁸ concentrating less on ethical analysis and more on social analysis.

In this paper we want to present a case-based approach that emphasizes the connection of ethical reflection and analysis to the practice of good software development. Central to this approach is the use of detailed historical cases. These cases can present the issues that software developers face in all their complexity, while emphasizing practice in the skills they will need to deal with that complexity. Cases of the complexity we present here are not readily available, though many of the ones presented by Spinello¹⁸⁹ and the cases available on www.computingcases.org can be used for this approach.

¹⁸⁵ See Chuck Huff and C. Dianne Martin, “Computing Consequences: A framework for teaching ethical computing,” *Communications of the ACM* 38 (December, 1995): 75-84. Also the ACM/IEEE Computing Curricula 2001 at <http://www.computer.org/education/cc2001/steelman/cc2001/index.htm>.

¹⁸⁶ Rob Kling (ed.), *Computerization and Controversy: Value Conflicts and Social Choices*, 2nd ed. (San Diego, 1996).

¹⁸⁷ Deborah G. Johnson, *Computer Ethics* (Englewood Cliffs, NJ, 2001), or Herman T. Tavani, *Ethics and Technology* (Hoboken, NJ, 2004).

¹⁸⁸ Chuck Huff and Thomas Finholt, *Social Issues in Computing: Putting Computing in its Place* (New York, 1994).

¹⁸⁹ Richard A. Spinello, *Case Studies in Information and Computer Ethics* (Upper Saddle River, NJ, 1997).

Cases can be used to accomplish all the goals for ethical education in computing:¹⁹⁰

- *Mastering a knowledge of basic facts and understanding and applying basic and intermediate ethical concepts.* Even simple cases can help students learn appropriate distinctions between various basic concepts (e.g., consequentialist vs. deontological approaches). They can also help students learn intermediate concepts (e.g., intellectual property, kinds of privacy, etc.). The cases do this while making it evident that the concepts are useful in real life, thus providing motivation for students. They also provide opportunities for students to practice skills such as applying a code of ethics and making and listening to ethical arguments.
- *Practicing moral imagination.* Cases, particularly complex ones, allow students to gain skill in the moral imagination they need to detect when an ethical issue is present, to understand how software affects people, and to take the perspective of others. They also allow students to practice the moral imagination they need to construct creative solutions to ethical problems (which involves trading off values, understanding organization politics, taking others' perspectives, etc.)
- *Encouraging the adoption of professional standards into the professional self-concept.* Cases that show the intimate connection between technical and ethical issues (like the Therac-25 case we present here) allow students to see that ethical issues cannot be separated from the technical issues they care about, and so encourage students to see ethical reflection as a part of their professional responsibility.
- *Building an ethical community.* Case discussion is an ideal venue for getting students used to talking with each other and influencing each other about ethical issues.

More than a traditional lecture or didactic approach, cases provide the opportunity for the manifestly social sorts of activities that the work in moral psychology suggests are important in acquiring the knowledge base and skills for moral judgment.¹⁹¹ When discussing, debating, constructing, or trying cases, students actively disagree with each other and influence each other's intuitions about what is and is not the right thing to do. They are not only learning explicit knowledge (like intermediate concepts or ethical skills) but also tacit knowledge, such as when to be suspicious, how to respond to disagreement, and so on. In this way, case-based pedagogy engages more of the systems/levels/processes of moral reasoning than do more passive instructional methods.

Taking various positions in a case can help students to learn moral sensitivity and to practice identifying moral issues from different perspectives. Moral imagination can be extended by wrestling with the complexities of a case and a variety of solutions. Simple self-reflection is often ineffectual because it tends to produce only rationalizations for existing moral intuitions; it does not provide reasons strong enough to motivate self-reform.¹⁹² But moral intuition can be influenced and shaped through our interactions with others. Hence, case discussion provides an ideal tool for promoting this interaction. Students put forth unexamined opinions and find that they fail to convince others; they listen to the views and reasons offered by others and, through this, learn to reexamine their own views from a different, broader perspective. Consequently, case discussion develops the habit of putting forth rational arguments for one's moral views.

¹⁹⁰ Chuck Huff and William Frey, "Moral Pedagogy and Practical Ethics," *Science and Engineering Ethics* (In press).

¹⁹¹ Ibid.

¹⁹² Jonathon Haidt, "The Emotional Dog and Its Rational Tail: A Social Intuitionist Approach to Moral Judgment," *Psychological Review* 108 (2001): 814-34.

Advantages of Historical Cases

But why use historical cases instead of the more neatly packaged cases that philosophers tend to use? Partly because the goals of a computer ethics course are different from the standard philosophical ethics course. Computer ethics courses emphasize the skills of detecting and framing ethical issues, taking the perspective of the other, making technically based ethical arguments, and constructing and proposing solutions to ethical difficulties. All of this needs to be done within the complexity of the sort of ethical situations that students are likely to face in industry. Historical cases provide the needed complexity.

But they also provide the uncertainties that allow students to practice constructing solutions to deal with contingencies. You can never know all you think you need to know, even in a very detailed case. And in a detailed case, it is clear from the detail that decision makers at the time were operating under uncertainty but still had to make decisions.

This connection with the real difficulties of professionals in the field is another benefit that historical cases offer. Students are motivated when they discover that they are learning knowledge and skills that will help them address similar difficulties when they find themselves in similar circumstances. Thus, historical cases need to be chosen and presented in a way that makes this connection to the difficulties students are likely to face in the modern workplace.

Lastly, historical cases are helpful because they provide real outcomes of real decisions. Students can compare their own skills and knowledge with those of the decision makers in the case, and see the actual effects of at least the one option the decision maker chose. In what follows, we present a classic historical case in the sort of detail needed for extensive class use and we discuss how we have used the case in class.

The Therac-25 Case

Therac-25 was a new generation medical linear accelerator introduced in 1983 for treating cancer. It incorporated the most recent computer control equipment. Therac-25's computerization made the laborious process of machine setup much easier for operators, and thus allowed them to spend minimal time in setting up the equipment. In addition to making setup easier, the computer also monitored the machine for safety. With the advent of computer control, hardware-based safety mechanisms were transferred to the software. Hospitals were told that the Therac-25 medical linear accelerator had "so many safety mechanisms" that it was "virtually impossible" to overdose a patient. As it turned out, the computerized safety monitoring was not sufficient, and there were six documented cases of significant overdoses of radiation, three resulting in severe injury and three resulting in death from radiation sickness.

The standard reference on the Therac-25 accident is the extensive paper by Leveson & Turner.¹⁹³ This paper documents the history of the development of Therac through several generations, the introduction of the machine, and the accidents it caused. It is particularly helpful in documenting the safety analysis done by the company and in showing in detail how the computer control was accomplished, even to including pseudo-code to document the software errors that produced several of the accidents. With support from the National Science Foundation (DUE-9972280 and DUE-9980786), we have been able to add to this analysis from

¹⁹³ Nancy Leveson and Clark S. Turner, "An Investigation of the Therac-25 Accidents," *Computer*, 26 (July 1993): 18-41.

legal documents we were able to collect, interviews we did with relevant actors (e.g., an operator of radiation therapy machines, a medical physicist), and other resources. Much of this is available on the computingcases.org website, but some is unique to this article.

A history of the introduction and shutdown of Therac-25

Therac-25 was released on the market in 1983 by Atomic Energy Canada, Limited (AECL). In 1987, treatments using the eleven machines then in operation were suspended. Those machines were refitted with the safety devices required by the FDA and remained in service. No more accidents were reported from the use of these machines. At about that time, the division of AECL that designed and manufactured Therac-25 became an independent company.

The major innovations of Therac-25 were the double pass accelerator (allowing a more powerful accelerator to be fitted into a small space, at less cost) and the move to more complete computer control. The move to computer control allowed operators to set up the machine more quickly, giving them more time to speak with patients and making it possible to treat more patients in a day. Along with the move to computer control, most of the safety checks for the operation of the machine were moved to software and hardware safety interlocks were removed.

The accident history of Therac-25

In July 1985, AECL was notified that a patient in Hamilton, Ontario, had been overdosed. AECL sent a service engineer to the site to investigate. AECL also informed the U.S. Food and Drug Administration (FDA) and the Canadian Radiation Protection Board (CRPB) of the problem. In addition, they notified all users about the problem and issued instructions that operators should visually confirm hardware settings before each treatment. AECL could not reproduce the malfunction, but its engineers suspected that a hardware failure in a microswitch was at fault. They redesigned the hardware and claimed that this redesign improved the safety of the machine by five orders of magnitude. After modifications were made in the installed machines, AECL notified sites that they did not need to manually check the hardware settings anymore.

In November 1985, AECL heard of another incident in Georgia. The patient in that incident (Katy Yarbrough) filed suit that month based on an overdose that occurred in June. There is no evidence that AECL followed up this case with the Georgia hospital. Though this information was clearly received by AECL, there is no evidence that this information was communicated internally to engineers or others who responded to later accidents. This lack of internal communication is likely the cause of later statements that there was no history of overdosing with the machine.

In January 1986, AECL heard from a hospital in Yakima, Washington, that a patient had been overdosed. The AECL technical support supervisor spoke with the Yakima hospital staff on the phone, and contacted them by letter indicating that he did not think the damage they reported was caused by the Therac-25 machine. He also notified them that there have "apparently been no other instances of similar damage to this or other patients."

In March 1986, AECL was notified that the Therac-25 unit in Tyler, Texas, had overdosed a patient. They sent both a local Texas engineer and an engineer from their home office in Canada to investigate the incident the day after it occurred. They spent a day running

tests on the machine but could not reproduce the specific error. The AECL engineer suggested that perhaps an electrical problem had caused the accident. This was based in part on the patient's report that the accident produced a "frying" sound from the machine and significant heat. The engineer also said that AECL knew of no accidents involving radiation overexposure with the Therac-25. An independent engineering firm checked out the electric shock theory and found that the machine did not seem capable of delivering an electric shock to a patient.

On April 11, 1986, AECL was alerted to another overdose that had occurred in Tyler. After communication with the medical physicist at Tyler, who had discovered how to replicate the malfunction, AECL engineers were able to reproduce the overdose and the sequences leading up to it. The memo "How to produce a malfunction 54" from the medical physicist is available on the computingcases.org website.

AECL filed a medical device report with the FDA on April 15, 1986 to notify them of the circumstances that produced the two Tyler accidents. At this point, the FDA, having been notified of the first Tyler accident by the hospital, declared Therac-25 defective and ordered the firm to contact all sites that used the machine, investigate the problem, and submit a report called a "corrective action plan." AECL contacted all sites and recommended a temporary fix that involved removing some keys from the keyboard at the computer console. In teaching the case, this is the point at which we "freeze action" in the case and ask students to play the role of a software developer at AECL and plan a response. It is significant that a later death from a different race condition occurs in the historical case.

The FDA was not satisfied with the notification that AECL gave sites, and in May 1986 required AECL to re-notify all sites with more specific information about the defect in the product and the hazards associated with it. At this time AECL was also involved in meetings with a "users' group" of Therac-25 sites to help formulate its corrective action plan. After several exchanges of information between AECL and the FDA (in July, September, October, November, and December 1986), AECL submitted a revised corrective action plan to the FDA.

In January 1987, AECL was notified of an additional overdose occurring at the Yakima, Washington hospital. After sending an engineer to investigate this incident, AECL concluded that there was a different software problem that allowed the electron beam to be turned on without the device that spread it to a safe concentration being placed in the beam.

In February 1987, the FDA and its Canadian counterpart cooperated to require all units of Therac-25 to be shut down until effective and permanent modifications were made. After another six months of negotiation with the FDA, AECL received approval for its final corrective action plan. This plan included numerous software fixes, the installation of independent, mechanical safety interlocks, and a variety of other safety-related changes.

Several of the surviving victims and families of deceased victims filed suit in U.S. courts against AECL and the medical facilities using Therac-25. All of these suits were settled out of court. More detailed information about the specific accidents and their outcomes is available on the website.

Description of the Therac-25 socio-technical system

The approach we use in describing cases is taken from Mumford (*The Myth of the Machine*, 1970). The central insight is that understanding the ethical issues associated with a technology requires understanding the technology and its use within a social system. Thus, the

socio-technical system is the mixture of people, technology, rules and procedures, data and data structures, and laws and regulations that structure the use of the technology and its effects on people. We provide a detailed look at the socio-technical system on the website, but give a more simplified version here.

The machine and software: There were two previous versions of Therac machines, each produced by AECL in collaboration with a French company, CGR. Therac-6 and Therac-20 (each named for the power of the beam they could produce) were based on earlier designs from CGR. By the time Therac-25 was released for sale, AECL had 13 years of experience with production of medical linear accelerators. Therac-25 was based on these previous versions. Its main innovations were: 1) a "double pass" electron beam so the machine could produce more energy in less space, and 2) the addition of extensive computer control of the machine. This latter innovation allowed AECL to move much of the checking for hazardous conditions into the software.

The Therac-25's ancestors, Therac-20 and Therac-6, had used a minicomputer (a DEC PDP-11) to add some convenience to the standard hardware of a medical linear accelerator. They both could work without computer control. AECL determined to make its new model, Therac-25, a tightly coupled combination of software and hardware. Therac-25 software was not written from scratch, but was built up from components that were borrowed from the earlier versions of Therac.

Therac-25 was a dual mode machine. This means that it could treat the patient with relatively low energy electron beams or with X-ray beams. This dual mode allowed for further cost savings in that two machines could be replaced by one. Therac-25 also had a "field light" position that allowed a standard light beam to shine in the path of treatment to help the operator in setting up the machine. Thus there were three modes in which the Therac-25 could operate: electron beam and X-ray for treatment, and field light for setup.

Even though they are relatively low energy, the electron beams are too powerful in their raw form to treat the patient. They need to be spread thinly enough to be the right level of energy. To do this, Therac-25 placed what are called "scanning magnets" in the way of the beam. The spread of the beam (and thus its power) could be controlled by the magnetic fields generated by these magnets. Thus, for electron beam therapy, the scanning magnets needed to be placed in the path of the beam. It was a race condition produced by a software error in setting the magnets and resulting in a turntable mismatch that produced at least two of the accidents.

X-ray treatment requires a very high-intensity electron beam (25 MeV) to strike a metal foil. The foil then emits X-rays (photons). This X-ray beam is then "flattened" by a device below the foil, and the X-ray beam of an appropriate intensity is then directed to the patient. Thus, X-ray therapy requires the foil and the flattener to be placed in the path of the electron beam.

The final mode of operation for Therac-25 is not a treatment mode at all. It is merely a light that illuminates the field on the surface of the patient's body that will be treated with one of the treatment beams. This "field light" required placing a mirror in place to guide the light in a path approximating the treatment beam's path. This allowed accurate setup of the machine before treatment. Thus, for field light setup, the mirror needed to be placed in the path where one of the treatment beams would eventually go.

In order to get each of these three assemblies (scanning magnets or X-ray target or field light mirror) in the right place at the right time, the Therac-25 designer placed them on a turntable. As the name suggests, this is a rotating assembly that has the items for each mode placed on it. The turntable is rotated to the correct position before the beam is started up. This is a crucial piece of the Therac-25 machine, since incorrect matching of the turntable and the mode of operation (e.g., scanning magnets in place but electron beam turned on high for X-ray) could produce potentially fatal levels of radiation. The original Leveson and Turner article includes diagrams of the machine and the turntable, as does the Web site.

Setup and actuation: The Therac-25 operator sets up the patient on the table using the field light to target the beam. In doing this, treatment parameters must be entered into the machine directly in the treatment room. The operator then leaves the room and uses the computer console to confirm the treatment parameters (electron or X-ray mode, intensity, duration, etc.). The parameters initially entered in the treatment room appear on the console and the operator simply presses "Return" to confirm each one.

The computer then makes the appropriate adjustments in the machine (moving the turntable, setting the scanning magnets, setting beam intensity, etc.), which takes several seconds. If the operator notices an error in the input parameters, he or she can, during the setup, edit the parameters at the console without having to start all over again from inside the treatment room. It was a race condition produced by editing of parameters and resulting in a mismatch of energy with turntable position that produced at least two of the accidents.

When the computer indicates that the setup has been done correctly, the operator presses the actuation switch. The computer turns the beam on and the treatment begins. The main tasks for which the software is responsible include:

- Monitoring input and editing changes from an operator
- Updating the operator's screen to show current status of machine
- Printing in response to operator commands
- Monitoring the machine status
- Rotating the turntable to correct placement
- Strength and shape of beam
- Operation of bending and scanning magnets
- Setting the machine up for the specified treatment
- Turning the beam on
- Turning the beam off (after treatment, on operator command or if a malfunction is detected)

The Therac-25 software is designed as a real-time system and implemented in machine language (a low-level language, difficult to read). The software segregated the tasks above into critical tasks (e.g., setup and operation of the beam) and non-critical tasks (e.g., monitoring the keyboard). A scheduler handled the allocation of computer time to all the processes except those handled on an interrupt basis (e.g., the computer clock and errors generated by computer hardware).

The difficulty with this kind of software is the handling of things that might be occurring simultaneously. For example, the computer might be setting the magnets for a particular treatment already entered (which can take eight seconds), while the operator has changed some of the parameters on the console screen. If this change is not detected appropriately, it

may only affect the portion of the software that handles beam intensity, while the portion of the software that checks turntable position is left thinking that the old treatment parameters are still in effect. These sorts of scheduling problems that occur when more than one process is running concurrently are called "race conditions," the primary cause of the accidents.

In 1983, just after AECL made the Therac-25 commercially available, AECL performed a safety analysis of the machine using Fault Tree Analysis. This involves calculating the probabilities of the occurrence of varying hazards (e.g., an overdose) by specifying which causes of the hazard must jointly occur in order to produce the hazard.

In order for this analysis to work as a Safety Analysis, one must first specify the hazards (not always easy) and then be able to specify all the possible causal sequences in the system that could produce them. It is certainly a useful exercise, since it allows easy identification of single-point-of-failure items and the identification of items whose failure can produce the hazard in multiple ways. Concentrating on items like these is a good way to begin reducing the probabilities of a hazard occurring.

In addition, if one knows the specific probabilities of all the contributing events, one can produce a reasonable estimate of the probability of the hazard occurring. This quantitative use of Fault Tree Analysis is fraught with difficulties and temptations, as AECL's approach shows.

In order to be useful, a Fault Tree Analysis needs to specify all the likely events that could contribute to producing a hazard. Unfortunately, AECL's analysis almost entirely left out consideration of the software in the system. Since much of the software had been taken from the Therac-6 and Therac-20 systems, and since these software systems had been running many years without detectable errors, the analysts assumed there were no design problems in the software. The analysts considered software failures like "computer selects wrong mode," but assigned them probabilities like 4×10^{-9} .

These sorts of probabilities are likely assigned based on the remote possibility of random errors produced by things like electromagnetic noise, or perhaps the mean-time-between-failures data generally available then for PDP-11 machines. They do not take into account the possibility of design flaws in the software. This shows a major difficulty with Fault Tree Analysis as it was practiced by AECL. If the only items considered are "failure" items (e.g., wear, fatigue, etc.) a Fault Tree Analysis really only gives one a reliability for the system.

Hospitals: The complexity of cancer treatment organizations is one of the things students must deal with as they struggle to understand why the accidents happened and to construct a plan to respond to the accidents.

Cancer treatment facilities are often housed in large hospitals, but some are stand-alone centers (like the one in Tyler, Texas). Those associated with hospitals are more likely to be non-profit, while those that stand alone are more likely to be for-profit organizations. Financial pressures are likely to be strong at both for-profit and not-for-profit organizations, but they will have slightly different regulatory structures.

During the time of Therac-25 (the mid 1980s) a well-equipped treatment facility might have three different machines. The machines would be capable of producing different kinds of radiation and different strengths of beam, as well as different kinds of exposure to the patient. Each of these machines alone would cost between \$1 million and \$2 million. In addition, each

machine would need special housing with shielding in the walls, an adequate power supply, video and intercom links, and so on.

Operators would be needed to run each machine. For larger facilities, a supervisor with more training and experience might be needed to oversee the operators. In addition, at least one MD specialist in cancer radiation therapy (a Radiation Oncologist) would be required. A medical physicist would be needed to maintain and check the machines regularly; some facilities contract out these services. Lastly, these specialists would require support personnel (nurses, secretaries, administrative staff, people to handle billing and paperwork, janitorial staff, etc.).

Medical Linear Accelerators do age over time, and older machines often produce more errors. Five to ten years is a reasonable life span for a machine. Thus, simply to maintain a set of three medical linear accelerators, an institution can expect to spend \$1 million to \$2 million every third year.

Sometimes errors can be resolved and a machine can be kept longer using software upgrades or upgrades or retrofits of machine parts. The companies that sell linear accelerators charge for maintenance contracts that can include different levels of support. Because of monetary constraints, sometimes facilities are forced to choose between software updates, manuals, and training for operators and physicists. All this is in addition to the price of the machine itself.

Production pressures are always present when an expensive medical technology is being used. These very expensive machines need to treat enough patients to pay for themselves over their lifetime. And in for-profit medical facilities, the additional pressure of generating a profit is added to this production pressure. Another kind of production pressure is generated because of concern for the patient. Patients' schedules require treatments on certain days, and it disrupts their lives and slows down their treatment to have to reschedule them while the machine is being checked out.

These production pressures generate the desire to "push patients through." If a machine gives only a portion of the prescribed dose, an operator will often repeat the treatment with enough radiation to add up to the total prescribed dose. Of course, because of liability issues and concerns for patient welfare, this can only be done when it is thought safe.

One of the advantages of the significant computerization of the Therac-25 machine was that setup for treatment could be done much more quickly. This allowed the operator more time to speak with patients and interact with them about their health concerns. In addition, this increased efficiency allowed more patients to be scheduled each day. Thus, more patients could be treated without the atmosphere being reduced to that of a factory.

Facilities that run medical linear accelerators are surely concerned about liability for patient injuries that might occur. Insurance for medical providers is quite expensive; errors in treatment can result in lawsuits, which in turn produce increases in insurance premiums. Standard practice in litigation is to "sue everyone with deep pockets." This means that even if an error is the result of the poor design of a linear accelerator, the facility itself will be sued simply because they were involved: they have insurance and, thus, "deep pockets."

But it is in the interest of facilities to reduce errors without the threat of lawsuits. When a treatment must be restarted several times because of errors, it may reduce patient confidence in

the facility. This can result in patients moving to another facility with which they are more comfortable. Lastly, medical professionals are in their business because they want to help people and have the knowledge and skills to do so. So a primary motivation of medical professionals is patient welfare.

The FDA: In addition to dealing with the technical issues and organizational issues associated with the hospitals, students need to consider the role the FDA plays in regulating medical devices. Understanding the constraints the FDA imposes on possible solutions (and the opportunities they provide) is a crucial part of designing a responsible solution to the accidents.

The Food and Drug Administration (FDA) was created when Congress passed the Food and Drugs Act in 1906. This act was the first of a series of laws and amendments that gave the FDA jurisdiction over the regulation of foods and patent medicines. In 1938, Congress strengthened and expanded the FDA to include the regulation of therapeutic and medical devices within its jurisdiction.

The FDA's Bureau of Medical Devices and Diagnostic Products was created in 1974, and soon operated in conjunction with the Medical Devices Amendments of 1976. The amendments helped to clarify the logistics of the regulation of medical devices, and required the FDA to "ensure their safety and effectiveness."

Radiation had been recognized as a health hazard since before World War I, and the FDA monitored the health risks that radiation-emitting products posed to America's workers and consumers. As FDA's responsibilities for monitoring radiological devices grew, a bureau within the FDA called the Center for Devices and Radiological Health (CDRH) was established.

In 1980, the FDA's budget had swelled to more than \$320 million, with a staff of more than 7,000. Many bureaus controlled areas such as biological drugs, consumer products, public health standards, and veterinary medicines.

FDA approved medical devices before they "went to market." This was called Pre-Market Approval and was a somewhat complex process. In the FDA Pre-market Approval scheme, devices were organized into three classes, as established by the 1976 Medical Device Amendments.

- I. Class I devices, "general controls provide reasonable assurance of safety and effectiveness," for example, bedpans and tongue depressors.
- II. Class II devices, such as syringes and hearing aids, "require performance standards in addition to general controls."
- III. Class III devices, like heart valves and pacemakers, are required to undergo pre-market approval as well as complying with general controls.

In addition to classifying devices as Class I, II, or III, FDA approved devices for market in one of two ways:

1. Proof of Pre-market Equivalence to another device on the market, termed 501(k); or
2. Pre-market Approval (Rigorous Testing).

If a company could show Pre-market Equivalence (proof that a new product was equivalent to one already on the market), the new product could be approved by FDA without

extensive, costly, rigorous testing. In 1984, about 94 percent of medical devices came to market through Pre-market Equivalence.

If a product was not equivalent to one that was already on the market, FDA required that the product go through testing to gain Pre-market Approval. In 1984, only about 6 percent of medical devices were required to go through this testing.

Thus, it was clearly in the interest of medical-device producers to show that their product had pre-market equivalence. The Therac-25, brought to market in 1983, was classified as a Class II medical device. Since AECL designed the Therac-25 software based on software used in the earlier Therac-20 and Therac-6 models, Therac-25 was approved by FDA under Pre-market Equivalency. This declaration of pre-market equivalence seems optimistic in that: 1) most of the safety mechanisms were moved into the software, a major change from previous versions of the machine; and 2) the confidence in the safety of much of the software was based on its performance in the older machines, which had hardware safety devices installed to block potential accidents.

A 1983 General Accounting Office (GAO) report criticized the FDA's "adverse experience warning system" as inadequate. FDA had published reports about potential hazards, including reports in their own newsletter, *The FDA Consumer*. The FDA implemented the mandatory medical-device reporting rule after Congress passed the Medical Device Reporting Legislation in 1984. This rule required manufacturers to report injuries and problems that could cause injuries or death.

Before 1986, users of medical devices (hospitals, doctors, independent facilities) were not required to report problems with medical devices. Instead, under the medical device reporting rule, manufacturers of these devices were required to report problems. The idea was that manufacturers would be the first to hear about any problems with the devices they made and that, therefore, reports would be timely. In addition, manufacturers would be most likely to have the correct information needed about a device to help resolve difficulties.

In the mid-1980s, the FDA's main enforcement tool for medical devices already on the market was publicity. The FDA could not force a recall, it could only recommend one. The CDRH (Center for Devices and Radiological Health, which monitors radiological devices) issues its public warnings and advisories in the *Radiological Health Bulletin*. Before issuing a public warning or advisory, the FDA could negotiate with manufacturers in private (and, in the case of Therac 25, with regulatory agencies in Canada). In response to reports of problems with a medical device, the FDA could, in increasing order of severity:

1. Ask for information from a manufacturer.
2. Require a report from the manufacturer.
3. Declare a product defective and require a corrective action plan (CAP).
4. Publicly recommend that routine use of the system on patients be discontinued.
5. Publicly recommend a recall.

Thus, even when the FDA became aware of the problem, they did not have the power to recall Therac-25, but only to recommend a recall. After the Therac-25 deaths occurred, the FDA issued an article in the *Radiological Health Bulletin* (Dec. 1986) explaining the mechanical failures of Therac-25 and explaining that the "FDA had now declared the Therac-25 defective, and must approve the company's corrective action program."

After another Therac-25 overdose occurred in Washington State, the FDA took stronger action by "recommending that routine use of the system on patients be discontinued until a corrective plan had been approved and implemented."¹⁹⁴ AECL was expected to notify Therac-25 users of the problem, and of FDA's recommendations.

After the Therac-25 deaths, the FDA made a number of adjustments to its policies in an attempt to address the breakdowns in communication and product approval. In 1990, health-care facilities were required by law to report incidents to both the manufacturer and the FDA.

AECL and the state of the technical art: A crucial part of the student's job in understanding the difficulty of the case is understanding the problems with synchronization of concurrent processes (explained in detail in the teaching section later). But they also need to understand what the programmers of the Therac-6, Therac-20, and Therac-25 were likely to have known about these issues. Almost nothing is known about the specific qualifications of the Therac-25 programmers (or even their identities), but we can get some idea of the current state of the art at the time.

Although software solutions to synchronization problems were known in the mid-1970s when AECL and CGR developed the Therac-6 software, it seems unlikely that those programmers would have used them in their implementation. An elaborate and complicated solution by Dekker¹⁹⁵ was available, but was "a tremendous mystification,"¹⁹⁶ difficult to comprehend or to program correctly. Strategies that depend on adding special features to the operating system appeared beginning with Dijkstra;¹⁹⁷ however, such operations did not appear as a standard part of common operating systems until years later, and their implementation in the Therac-6 system seems unlikely unless they had a specialist on their team. Operating systems courses at the time focused on theoretical discussions rather than practical implementation issues. John Lions' famous commentary on UNIX Version 6 source code (Lions, 1996), developed in 1975-76 for his students in Australia, is widely considered as the first operating systems course text to address practical concerns seriously (Tanenbaum, 1987).

Thus, assembly programmers in the mid-1970s would undoubtedly have employed hardware solutions for synchronization, if they were even aware of subtle issues such as race conditions. A "test and set lock" (TSL) instruction provided one approach, in which a particular machine instruction had the capability to copy a shared variable's value to another location held privately by a task, and to assign a value to that shared variable, all in a single indivisible operation. Other machine instructions (e.g., SWAP) could serve the same purpose. However, an examination of the PDP-11 instruction set¹⁹⁸ shows that no such instruction exists on the machine used for the Therac systems. It is conceivable that the "subtract one and branch" (SOB) instruction, designed for loop control, might have been turned to this purpose by a creative and careful programmer who had awareness of these synchronization issues.

These facts hardly exonerate the Therac programmers. The operating-system level routines for Therac-25 were written specifically for that system according to Leveson and Turner; those programmers had a responsibility to know about issues such as race conditions in

¹⁹⁴ *Radiological Health Bulletin* (March 1987).

¹⁹⁵ Cited on p. 58 of E.W. Dijkstra, "Co-operating Sequential Processes," in F. Genuys (ed.), *Programming Languages*, Academic Press (London, 1968).

¹⁹⁶ *Ibid*, 66.

¹⁹⁷ *Ibid*.

¹⁹⁸ Available at: <http://www.village.org/pdp11/faq.pages/PDPinst.html>.

multitasking real-time systems. They would most likely have heard about postponements of the release of the IBM 360 time-sharing system and of Multics. Both of these projects were late in part because of the difficulty of getting synchronization done properly. However, unless those responsible for the operating-system executive routines had prior experience writing concurrent software, it seems quite conceivable that they had never seen the subtleties of race conditions.

Teaching With the Therac-25 Case

We use the Therac-25 case as the primary case in the required *Ethical Issues in Software Design* course in the computer science major at St. Olaf College. Students also use eight to ten other cases during the course, but they learn Therac-25 first and learn our approach to cases this way. Two CS courses (Intro CS and *Software Design*) are prerequisites for the ethics course, so students know something about computer science and software development methods. In these prerequisite courses, students are introduced to our emphasis on socio-technical systems and do a preliminary ethical analysis of a piece of software, including stakeholder analysis and the identification of ethical issues. Thus, students come to the *Ethical Issues* course prepared to discuss a complex case like Therac-25 in considerable detail and with some technical sophistication.

In addition to discussing cases, the main project in the *Ethical Issues* course is a semester-long social and ethical analysis of an existing or proposed socio-technical system. This year, for instance, students in teams analyzed a proposed new data system for the school's registrar. The project requires interviewing people involved, analyzing documentation, collecting relevant data (by observation, interface mock-ups, questionnaires, etc.), and using a structured approach to describing the socio-technical system and the ethical issues it raises. As seniors, students know they will be required to use these skills to analyze their senior projects. Thus, they know that the skills they are learning in discussing Therac-25 will help them complete their class project and later coursework in the major. One student has recently taken a job with a firm upon graduation to do projects that are substantially like those in this course. The important point here is that discussion of the historical Therac-25 case is embedded in a series of requirements and opportunities that make the skills the student learns from the case quite relevant.

Prior to beginning the case, students read about socio-technical analysis and ethical analysis and learn methods for doing both. This takes several weeks and uses simplified historical cases. Students then read the case material on the website before the first class period, and are given questions that they will be expected to answer in that period. Presentation of the socio-technical context of the case and the liability issues take up this first class period. A second class is used to analyze the pseudo code and present the technical issues (see the sections below). In a final class period, students are then asked to role play an in-house AECL software developer at the time AECL submitted its first corrective action plan (CAP). The assignment is to decide how to proceed from that point.

A crucial place in the decision stream

The problem that students are posed can be succinctly stated: does implementing the CAP solve the problem? Because we know the outcome of the case, we know the answer is "No." But since they are in the stream of decision, in the middle of the case, they need to argue only from the information they have available and their current expertise. They need to propose a solution that works for AECL and its customers and that can be "sold" inside AECL as a feasible approach.

By this time, students certainly feel they have voluminous evidence in front of them, but still they long for more specifics. So the first step is to outline what additional information it would help to have. Answers to some of these questions are available from the website (e.g., the timeline of the accidents; were there consultants available at the time who specialized in safety analysis?). Other answers can be estimated from what we know (e.g., ratio of number-successfully-treated to accidents). Others are simply unknowable at the time (e.g., are there any other software errors?).

The next step is, in the tradition of ethical analysis, to list the relevant stakeholders. These are the people who need to be satisfied with any proposed solution. Then we list decision options (e.g., do nothing and declare the CAP the solution, recall all Therac-25s immediately, suspend treatment for a specified time period to study the software for errors, etc., etc.) and we look for the effects of each of these decisions on the various stakeholders. Then we look for variations or combinations of the decision options that we think we can sell inside AECL and that best serve the interests of the stakeholders. After agreeing on a solution, we then make a plan for how we would sell the solution and what our options would be if we cannot get our solution implemented in a satisfactory manner.

This year, for instance, our students decided that since eliminating one software error did not assure them there were no errors left, the best plan was to hire outside safety consultants to review the software and the system. This would require, they felt, notifying the treatment centers that such a review was under way and giving them the option of suspending treatment during that time. The length and expense of the review would be negotiable with the company, but students felt they would need to protest vigorously if no review was done or if hospitals were not notified (there was some disagreement on this). Early warning reports from the review would help hospitals decide whether to suspend treatment or reinstate it as the review progressed. But they were insistent that the review needed to be limited in time (from three weeks to several months) to limit the costs (to AECL, the treatment centers, and the patients) of not treating patients during that time.

Going through these steps together gives students considerable practice in both recognizing ethical issues associated with a computing system and in coming up with creative solutions to resolve them. Students are also introduced to the idea of ethical dissent if their solutions are not adopted. Later cases in the class look at ethical dissent in more detail, but this case becomes a touchstone for when it might be required.

A beginning technical lesson from the case

One of the software errors was documented as the cause of a death at the Yakima, WA treatment center. The error is simple enough for students with little CS background to understand, though it raises profound ethical issues about the reuse of software. This error involved incrementing a shared variable called Class3 instead of doing a Boolean. Class3 indicated whether a particular piece of equipment that shaped and limited the beam (the collimator) was or was not in the right position. If the collimator's position was correct, Class3 was set to zero; otherwise it was set to some non-zero integer by the simple expedient of incrementing it. Class3, however, was stored in a single byte, and so every 256 times it was incremented, it would overflow and register zero. If the operator pressed the set key at precisely that moment, the machine would schedule a treatment without the shaping device in the correct

place, causing a race condition which resulted in an overdose. This occurred rarely enough that it was difficult to detect.

The contemporary milieu in which Therac programmers worked informs our understanding of the Yakima incrementation error. Incrementing a variable to change its state from zero (FALSE) to non-zero (TRUE) was a common and standard practice in the day. Assembly language programmers particularly made use of such tricks to save a few precious CPU cycles in their code. The problem with incrementation to change a Boolean variable arises when that variable overflows. In the Yakima problem, such overflows, together with bad timing (race conditions), had fatal consequences. Using a two-byte PDP-11 word instead of a single byte for that variable would have made these overflows 256 times less likely (one in 65,536 instead of one in 256); a four-byte longword might never have overflowed (one in over four billion). But longwords were not available in the “16-bit” PDP-11 architecture, unless one built a multi-precision integer variable oneself (at a cost to performance); and the first inclination of any assembly programmer would be to try to get away with a single byte, applying the same minimal-usage philosophy to memory space as to CPU execution time.

Furthermore, it is unclear whether the requirements for the early 1970s (and more simple) Therac-6 software influence the decision to use incrementation instead of a Boolean. Was incrementation of an 8-bit variable to change its (Boolean) state part of the reused Therac-6 software, perhaps intended for initial calibration only, but later used throughout the control of the more complicated dual mode Therac-25? We cannot tell without knowing more specifics about the Therac code and its evolutionary development. In any case, this hypothesis indicates the kind of imagination about possible effects when a bit of software is reused that we must expect from programmers. Such imagination is difficult enough to instill in current-day programmers, and was in very short supply among assembly language programmers of the 1970s.

The world has learned much about how to think about software reliability over the past three decades. It is now inconceivable that the FDA would approve a software-dependent device such as the Therac-25 with as little testing of the software functionality as that system received—due in large part to the experience gained from this very system and others that looked acceptable at the outset, but later generated unanticipated problems. The word that little logical flaws such as incrementing to change Boolean state could result in catastrophic failure was slow to trickle back to programmers.

Students who are introduced to this “technical” issue in the Therac-25 case are made aware of how important documentation is when software may be reused. They become sensitized to the ethical issues that can arise from shifting requirements as software is developed. They begin to understand how one piece of software in one socio-technical context can operate flawlessly and, in another context, can produce significant harm. And they begin to become aware of what “best practices” mean, how they can shift over time, and why they are important. All of these points make it clear how closely coupled technical and ethical issues are.

An advanced technical lesson from the case

The other documented software error was the cause of two deaths in the Tyler, TX treatment facility. It involved concurrent access to a shared variable that set up a race condition resulting in a mismatch of beam energy with the turntable position. In this case, it involved monitoring and verifying treatment data input from the operator. The variable indicated whether data entry was completed or not. Under certain conditions, a race condition would

occur with this variable. If the operator set initial parameters and later edited them, the later edits would show on the screen but would not be detected by the portion of the software that implements the machine settings. This could result in the wrong piece of the turntable being in place with the high energy beam turned on, producing a massive overdose.

In technical terms, these race conditions would have been resolved by indivisible set and test operations. Explaining what this means and why it is important is a topic that comes up in operating systems courses today, but that was not widely known and only poorly understood in the early 1970s when some of the Therac-6 software was produced. But students today can see that technical issues from their courses can have profound ethical implications.

A closer look at the staged development of the Therac-6, Therac-20, and Therac-25 software, given “best” computing practices in the 1970s, raises uncertainties about the overall responsibility for race condition failures in that system. In order to understand these uncertainties, students first need a review of (or introduction to) the technical issues.

A race condition in a software system arises when the correct operation of that system depends on timing of execution. Race conditions can only occur in systems that involve multiple tasks (or processes) that carry out their instructions concurrently, or at the “same time,” perhaps through a time-sharing operating system. To illustrate the subtle nature of race conditions in a multitasking system, consider Dijkstra's classic “dining philosophers” problem (1965): Imagine several individuals sitting around a dinner table, each of whom alternates between thinking and eating for indeterminate amounts of time. In order to eat, one of these “philosophers” requires two utensils. In Dijkstra's analogy, two forks were required to eat spaghetti; others have suggested chopsticks and Chinese food. Unfortunately, each philosopher must share his/her utensils with his/her neighbors. Imagine a circular table with eight plates around the edge and eight single chopsticks, one to the right of each plate. To eat, each philosopher must pick up one chopstick from each side of his or her plate. This means that the philosophers on either side of the dining philosopher will have to wait to eat (in the analogy, they spend the time thinking), since they each have access to only one chopstick. In this hypothetical problem, the philosophers represent multiple concurrent tasks, and the chopsticks represent computing resources (variables, input/output devices, etc.) that those processes must share.

Correctly instructing (programming) each philosopher-task about how to proceed is a non-trivial problem, as the Appendix shows. For example, in an attempted solution algorithm for which each philosopher receives a Boolean-valued shared variable that is TRUE when that philosopher may safely pick up both utensils and eat, inopportune timing may lead one philosopher A to yield eating rights to a second philosopher B between the time when B checks for eating rights and the time when B begins to think, awaiting a “wakeup” from A; if A sent B's “wakeup” signal before B has begun to sleep, then B might enter a state of permanent inactivity while unknowingly holding the sole right to eat. Thus, the correct behavior of the system of philosopher algorithms depends on timing—a race condition. Naïve efforts to repair this approach by saving “wakeups,” sharing different variables, and other means simply move the bad-timing condition to another location in the multitasking algorithm. Furthermore, in practice, race conditions are among the most difficult system bugs to find, since one cannot duplicate the precise timing conditions that cause the transient error except by chance, unless one already knows what that transient error is. In the Therac-25 case, the Tyler race condition did not show up unless particularly well-practiced operators made editing changes very quickly.

The only effective way to combat race conditions is to avoid them. In the example above, one can avoid the lost “wakeup” by insuring that philosopher B cannot be interrupted between

the step of checking its shared variable and the step of entering its thinking phase, that is, by causing the system to perform both steps in a single indivisible operation. In this way, the philosopher is assured that the unfortunate timing error of having the wakeup call occur right after checking for it but before entering the thinking stage (when it would cause some action) is made impossible.

Now that we know what a race condition is, we can see in the Therac-25 how it can kill people. Therac-25 appeared in 1983, and its programmers modified the software from the earlier Therac-6 system that was programmed in the early 1970s. Given the era when software for these systems appeared (see the “state of the technical art” section earlier), it seems unlikely that machine language programmers in 1972 would have sufficient knowledge to know how to avoid race conditions with concurrent processes. This makes the technical error more understandable. But it brings up the larger issue, again, of documentation of code when it is reused and of the effect of shifting requirements and socio-technical systems. It also makes clear the importance of including a carefully designed maintenance phase in the software life-cycle; part of the reason the errors were not caught is that there was no clear mechanism for them to filter back to the appropriate people at AECL (or the FDA). Again, what appear to be technical issues become ethical ones. Students who learn about race conditions from the Therac-25 case know about the importance of a professional’s responsibility to be aware of best practices and the state of the art in one’s area of expertise, and the dangers of operating outside of their area of expertise.

Leveson and Turner say that: “focusing on particular software bugs is not the way to make a safe system .The basic mistakes here involved poor software engineering practices and building a machine that relies on the software for safe operation.”¹⁹⁹ This is not the lesson that is often drawn from the case, but presenting the case in this fashion makes it clear that technical decisions made by the programmers in the context of their historical environment had profoundly ethical implications for the eventual users of the system. In this way, putting the case in historical perspective makes it clear that ethical decision making is inseparable from good software design methodology, and that *good* in this context deserves its double entendre.

Relevance of Therac-25 to Other CS Courses and Concepts

Instructors can readily make connections between suitably detailed historical cases and many courses in the CS curriculum because these cases have enough complexity to touch on substantial topics from those courses. For example, the Therac-25 discussion in this paper includes significant content from at least the following undergraduate CS courses (in addition to courses specifically in Computing Ethics).

- **Computer Organization:** instruction set architecture; assembly language programming.
- **Operating Systems:** synchronization (interprocess communication), including race conditions; hardware and software solutions; multitasking operating system design.
- **Software engineering:** reuse; requirements analysis; software evolution and maintenance; ethical and professional context.

¹⁹⁹ Op. cit., Leveson and Turner, p. 38.

- **Elective courses:** History of Computing (historical context of computing at the time of Therac software development); Real-time Systems (example of small dedicated system, importance of correct synchronization); Systems Programming.

In each of these examples, the use of an historical case also connects course content to the complexity of problems in the applied world in terms of computing technology, ethical issues, and professional concerns.

These cases highlight pervasive themes that appear throughout the discipline of computer science. For example, taking the “recurring concepts” of the joint ACM/IEEE recommended computing curriculum.²⁰⁰ The Therac-25 example relates to at least the following:

- **Trade-offs and consequences:** efficiency vs. correct synchronization; productivity vs. quality; computing vs. non-computing solutions.
- **Reuse:** advantages of reuse; dangers of changing requirements; testing of systems with reused code; importance of documentation including specification of reusable code.
- **Evolution:** example of evolutionary system development in industry; risks of poorly managed evolutionary development; evolution of both code and requirements; resiliency of a system in the face of change.
- **Consistency and completeness:** formal methods; correctness; importance of proper system specification and adequacy of specifications; reliability practices for real-time systems; behavior of a system under error conditions and unanticipated situations.
- **Efficiency:** historical changes in attitudes towards software efficiency; importance of efficiency in context; efficiency of production in an applied project.
- **Security:** defense of a system against unanticipated requests; response of a socio-technical system to catastrophes; misuse of data types; user interface features; physical security measures; role of software in a security strategy; testing of system security.

Conclusion

To reiterate a claim made earlier, these connections of computer science principles to the historical case of the Therac-25 make it clear that good software development deserves the double entendre in the word “good:” that the ethical practice of computing requires deep technical knowledge and skill and commitment to good practice. This becomes most clear when we see historical cases of ethical issues in computing that have enough complexity to include technical detail. Dining philosophers help students understand what race conditions are; Therac-25 helps students understand why they matter.

²⁰⁰ Available at: <http://www.computer.org/education/cc1991/>.

Chuck Huff is Professor of Psychology at St. Olaf College. He has published research in the areas of moral reasoning, computing and education, gender and computing, social aspects of electronic interaction, and ethics in computing. He recently returned from sabbatical doing empirical research on the moral development of computer professionals in collaboration with the *Centre for Computing and Social Responsibility* at Demontfort University in Leicester, UK. *Author's address:* St. Olaf College, 1520 St. Olaf Ave., Northfield, MN 55057-1098 USA. Email: huff@stolaf.edu.

Richard Brown is Associate Professor and Director of Computer Science at St. Olaf College. He recently designed the college's computer science major in a way that integrates ethical issues throughout the curriculum. He is writing an introductory textbook, *Principles of Computer Science*, which emphasizes recurring concepts in computing and is structured around three programming paradigms: functional, imperative, and object-oriented. *Author's address:* St. Olaf College, 1520 St. Olaf Ave., Northfield, MN 55057-1098 USA. Email: rab@stolaf.edu.

Appendix A: Basics of Concurrency and Race Conditions

Multitasking

- A *process* (or **task**) is an execution of a program. Note that in a multi-user system, we would expect multiple processes executing various programs; it may be that two or more processes even execute the same program “simultaneously” through time sharing.
- **Multitasking** means using multiple tasks in a single system.
Example: Dining philosopher's problem (see below), which involves multiple tasks sharing various resources.
- The use of multitasking leads to new and complicated kinds of computer bugs.
Example: **deadlock**-existence of a set of processes, each of which is **blocked** (unable to run) waiting for an event that can only be caused by another process in that set. See first attempted solution of dining philosopher's problem below.
- Multiple processes may require **shared variables** in order to carry out their work.
Example: Two processes may use a shared “buffer” data structure for holding items that have been produced by one process but not yet consumed by another.

Example: Dining Philosophers Problem (E. Dijkstra, 1965)

- N processes share N resources in an alternating circular arrangement. Each process has two states (computing, interacting with resources); each needs exclusive access to its two resources while interacting with them.
- Dijkstra's statement: N philosophers sit down together to eat spaghetti. Each philosopher spends his/her time alternately thinking then eating. In order to eat, a philosopher needs two forks; each fork is shared with exactly one other of the philosophers. What procedure will allow all the philosophers to continue to think and eat?
- Algorithm 1. Each philosopher does the following:

```
repeat forever
  think
  pick up left fork (as soon as it's available)
  pick up right fork (as soon as it's available)
  eat
  return left fork
  return right fork
```

Issue: Deadlock occurs if each philosopher picks up his/her left fork simultaneously, because no philosopher then can obtain a right fork.

- Algorithm 2. Each philosopher does the following:

```
repeat forever
  think
  repeat
    pick up left fork (as soon as it's available)
    if right fork is available
      pick up right fork
    else
      return left fork
  until both forks are possessed
  eat
  return left fork
  return right fork
```


Strategies for correct IPC

- Race conditions are problems in *interprocess communication (IPC)* or **synchronization**, i.e., mechanisms and practices for creating correct systems having multiple processes.
- To prevent race conditions, one needs some kind of *atomic* or **indivisible** operations that leave no possibility for timing "gaps."
- An Operating Systems course examines several (equivalent) higher-level software strategies for correct IPC, including *semaphores*, *monitors* (cf. Java's `synchronized` objects), and *message passing*.
Comment: Therac-25 assembly programmers would not have any of these higher level solutions available unless they built them themselves--an unlikely scenario for time-pressured programmers with only low-level programming resources who might not even have awareness of the issue, especially given the care that correct programming of one of these strategies would require.
- There are also hardware solutions, such as having a *test and set lock (TSL)* instruction in the ISA (machine language) of the machine being used. In a TSL instruction, a memory value can be retrieved ("tested") and a new "lock" value substituted in a single indivisible machine instruction, preventing a "gap" between "testing" and "locking." Thoughtful use of a TSL instruction can correctly solve IPC problems.
- Leveson and Turner's analysis: *"It is clear from the AECL documentation on the modifications that the software allows concurrent access to shared memory, that there is no real synchronization aside from data stored in shared variables, and that the "test" and "set" for such variables are not indivisible operations. Race conditions resulting from this implementation of multitasking played an important part in the accidents."*
- *Comment: The Therac-25 system used a standard, highly regarded computer produced by DEC (Digital Equipment Corporation) called the PDP11. The PDP-11 instruction set includes no TSL instruction. (There is also no SWAP instruction to interchange values of two independent memory locations, which could serve in place of a TSL instruction.) Thus, Therac25 programmers would have had to devise something else for correct synchronization.*

V. Resources for Instructors

26. Key Resources in the History of Computing

Thomas Haigh
University of Wisconsin-Milwaukee

Abstract

This guide provides an annotated and highly selective list of specific online, published, and institutional resources for the history of computing. It focuses on English-language secondary sources. Online resources include institutional sites and a smattering of sample sites presenting virtual exhibitions or primary sources. The guide to published books and articles on the history of computing includes sections on overall and reference works, as well as the history of: computer hardware and the computer industry; programming languages and software; communication and control systems; business and administrative applications; scientific computing; and computer science. The guide also lists home pages and journals with history of computing content.

This paper provides a kind of annotated bibliography of resources for the history of computing. When preparing my introduction to the history of computing for the computer scientist, I soon found that while there are many, many Internet pages full of links to history of computing websites there is no one-stop place where an interested reader could learn about the main electronic, institutional, and printed resources in the field. In particular, very few of the Internet sites refer readers to printed books and papers (still the best sources of information on most areas of computing history) or exercise much quality control. So while this resource list makes no attempt to be exhaustive, it has the advantage of covering journals, museums, professional associations, books, and journal articles as well as the key websites.

There are, however, some definite limitations of scope to this guide. Most importantly, the list is biased toward secondary sources (basically books and articles written after the events concerned, usually by historians rather than the people involved) rather than primary ones (the original documents and records). It is also biased toward work produced by professional historians rather than journalists, and toward analytical work rather than personal memoirs. The Internet is full of websites produced by fans, hobbyists, and collectors and almost none of these are referenced here. So far, such sites are the best sources of factual information on the history of microcomputers and videogames, both topics I have almost ignored here. Emulators for mini- and microcomputers are readily available, and these can be excellent teaching aids. Unfortunately, emulation and retro computing sites tend to appear and disappear too rapidly to track. Because I read only one language, I also confined this to sources written in English, which in turn are mainly confined to the United States, Britain, and occasionally Canada.

There is also almost nothing here on useful sources for other kinds of social investigation of computing, such as computer ethics, courses on “cyber culture” and the sociology of the Internet, or books on the way computers have changed work life or society. Many of the most important sources for the professional historian of computing do not directly relate to computers at all, but are to be found in social history, business history, labor history, organizational sociology, and so on. I kept the brief simple: to identify the essential secondary information sources concerned directly with the history of computing.

To help those with little knowledge of the field, I have marked a small number of particularly important institutions and unusually accessible and broadly relevant books with the phrase “(Start here).”

On-Line Resources: Institutions

alt.folklore.computers (newsgroup)

(<http://groups.google.com/groups?q=alt.folklore.computers&hl=en&safe=off&meta=site%3Dgroups>)

In the absence of any computer history newsgroups, active listservs or web forums, the next best thing is this long-running and much-trafficked newsgroup where old-timers hang out and remember the good old days. Passionate discussions, hacker culture, reminiscences, and historical trivia. The Google news archive listed above is a great resource.

ASIS Special Interest Group on the History and Foundations of Information Science

(<http://www.personal.kent.edu/~tfroehli/sighfis/sighfis.html>)

ASIS (or ASIS&T as it now is; they haven't updated this site to reflect that) is the scholarly/professional society for information scientists, which is what technically and scientifically oriented library-type people call themselves. There are no comparable interest groups active within ACM, IEEE CS, or any historical societies.

Charles Babbage Institute (Start here)

(<http://www.cbi.umn.edu/>)

There are two main reasons to visit the Babbage website. The first is to access its collection of oral history interviews, now numbering several hundred. Almost all of these have been transcribed, and the transcripts can be downloaded. A simple Web interface lets you search the abstracts and keywords for names or phrases of interest. The other main reason is to browse the finding aids for its archival collections. A finding aid is basically a catalog listing the contents of each folder stored within the archives. While almost none of the archival material is available online (the exception being a selection of photos from the Burroughs collection), having access to the finding aids lets you see what relevant materials CBI holds, request materials in advance of a visit, or even purchase copies of particular documents without visiting. The site also holds the full run of the institute's newsletter, covering the history of computing since the late 1970s.

Computer History Museum (Start here)

(<http://www.computerhistory.org/>)

Although the museum is now settling into its new home, it will be some time until its exhibits are finished. For this reason, it has relied on its website in recent years, creating a number of on-line exhibitions based around timelines and photographs, and covering topics such as the Internet and the microprocessor. It also documents the museum's busy schedule of talks by computer celebrities, and hosts its glossy newsletter CORE.

IEEE Annals of the History of Computing (Start here)

(<http://www.computer.org/annals/>)

Few libraries have a full run of this journal. Fortunately, the IEEE Computer Society Digital Library now covers all 25 volumes in .pdf form, though the earlier issues seem to have been scanned without compression, resulting in some very bulky files. For recent volumes, there is a convenient feature to download the whole issue in a single file. Even if you do not have a personal subscription, your institution may have one.

National Archive for the History of Computing

(<http://www.chstm.man.ac.uk/nahc/>)

Similar to the Charles Babbage Institute, but for the UK. Its main content is catalogs of archival holdings stored at the University of Manchester.

Resource Center in Cyberculture Studies

(<http://www.otal.umd.edu/~rccs/>)

has an excellent collection of links, book reviews, and literally hundreds of syllabi concerned with cultural and sociological aspects of the Internet. Courses with historical content make up an appreciable minority of the online syllabi.

Software History Center

(<http://www.softwarehistory.org/>)

On-line resource, including a list of early mainframe software figures and a number of short anecdotes.

Virtual Museum of Computing

(<http://vmoc.museophile.org/>)

This used to be the most complete collection of links to pages on specific topics such as computer science pioneers and corporate histories. Unfortunately this site is getting progressively less useful because updates have been very scarce in recent years, many of the links have expired, and a number of mirrors (to which you may find yourself automatically redirected) have vanished.

Online Resources: Virtual Exhibitions and other Primary Sources

Classic Computer Magazine Archive

(<http://www.atarimagazines.com>)

This volunteer site holds the digital full text of a growing number of computer hobbyist magazines from the late 1970s and 1980s, including *Antic* and *Creative Computing*.

DigiBarn Computer Museum

(<http://www.digibarn.com/>)

One of those sites full of disorganized material, some of which scrolls down for page after page. It documents a physical homebrew museum which, like several such establishments across the US, seems to take a nineteenth century, "cabinet of curiosities" approach, cramming a lot of stuff into limited space. As well as pictures of rare and not-so-rare personal computers and peripherals, it hosts some interesting stories, some digitized documents and memorabilia, and lots of photographs of old software. There are huge numbers of sites around the net aimed at collectors, reproducing old documentation, photographing obscure prototype, and celebrating beloved machines.

Ed Thelen's Antique Computer Site

(<http://www.ed-thelen.org/comp-hist/index.html>)

Holds an unmatched trove of vintage computer documentation in its "on-line documents" section, including original manuals for several key computers and IBM punched card machines and a series of reports from the 1950s and 1960s surveying the early computer industry. This is a personal home site, hard to navigate but full of interesting nuggets.

Historic Documents in Computer Science

(<http://www.fh-jena.de/~kleine/history/>)

Someone called Karl Kleine has scanned a set of early manuals for programming languages, from FORTRAN to C++.

IBM Archives

(<http://www.ibm.com/history>)

Includes a catalog of their main archives. While most of the actual content is not on-line, they do appear to be working to add material –including a useful A-Z glossary and some materials concerning the PC and the 701, IBM's first big computer.

Smithsonian Computer History Collection

(<http://americanhistory.si.edu/csr/comphist/>)

Includes a guide to its Information Age exhibition on display in Washington, D.C., as well as on-line resources and oral histories.

The Mouse Site

(<http://sloan.stanford.edu/MouseSite/>)

Produced by historian Alex Pang, then working at Stanford library, this site includes a mass of information about the history of the mouse, from Doug Englebart's original designs to its commercialization and simplification for the Apple Macintosh. It includes a selection of Englebart documents on-line, interviews with key participants and rare video footage.

The NATO Software Engineering Conferences

(<http://www.cs.ncl.ac.uk/people/brian.randell/home.formal/NATO/index.html>)

Brian Randell, one of the editors of the proceedings of the celebrated Garmisch and Rome conferences at which the idea of Software Engineering was given its first serious discussion, has placed the original reports on-line together with some photographs and reminiscences.

UVA Computer Science: Computer Museum

(<http://www.cs.virginia.edu/brochure/museum.html>)

Apparently based on a physical collection of machines and memorabilia. Doesn't give much context, but does include a whole lot of nice scans of important computers, components, manuals and specification sheets (click on images from the main page for more or bigger ones).

World of Spectrum

(<http://www.worldofspectrum.org>)

I'll pick just one of the sea of retro computing, nostalgia and emulation sites. This one covers the much loved Sinclair Spectrum from the early 1980s, and links to dozens of emulators, an archive of thousands of programs, documentation and, most remarkably, a searchable database covering thousands of pages from old computer magazines. No professionally trained historians have focused yet on this era, but there is certainly a mass of source material out there.

Key Books and Articles on the History of Computing

The main unit of scholarly production in history is the book. Here is a selection from the enormous number now produced. Emphasis is on scholarly rather than journalistic work. Everything included here is either excellent or the best work in an important subject area. For topics where no good book is yet available, and to interesting work not yet available in book form, a few papers have also been included.

Reference Works and Overall Histories

Martin Campbell-Kelly and William Aspray, *Computer: A History of the Information Machine* (New York, NY, 1996). **(Start here)**

The best overall history of the computer. It examines at the computer primarily as an extension of other business tools, focuses on business history of the firms producing computers. Summarizes a lot of earlier work while weaving it into a story. Good treatment of ENIAC and its commercialization as Univac, IBM's mainframes of the 50s and 60s and the first big on-line applications. A second edition is due in 2004.

Paul E. Ceruzzi, *A History of Modern Computing* (Cambridge, MA, 1998).

This is the other main history of the computer. More technical in its approach than Campbell-Kelly & Aspray, with less emphasis on business application and continuities with pre-computer technologies. However, it does feature good treatment of computer architecture and DEC – makes a strong case for the minicomputer as the precursor of modern personal computing. A second edition recently appeared, strengthening coverage of the 1990s.

Software History Bibliography (Charles Babbage Institute).

<http://www.cbi.umn.edu/shp/bibliography.htm>.

The online CBI software history bibliography includes references to monographs, journal articles, reports, oral histories, and archival collections relating to the history of software.

J.A.N. Lee, *Computer Pioneers* (Los Alamitos, CA, 1995).

Biographies and obituaries of leading figures. Currently out of print and almost impossible to find second hand, though a library near you probably has a copy.

Michael R. Williams, *A History of Computing Technology* (Englewood Cliffs, NJ, 1985).

Well produced, technical history of computation from scientific perspective. There is now a second edition.

Computer Hardware and the Computer Industry

Charles J. Bashe, et al. *IBM's Early Computers* (Cambridge, MA, 1986).

Written by a team of IBM insiders, this history is exhaustively researched, detailed and clearly written. There is no better guide to the computers of the 1950s. Currently out of print. A companion book, *IBM's 360 and Early 370 Systems*, by several of the same authors, covers similar ground for the 1960s.

Ross Knox Bassett, *To the Digital Age* (Baltimore, 2002).

Most books about the personal computer era have tended to take the underlying hardware technologies for granted. This book describes the development of MOS transistors, their integration onto silicon chips, and the creation of the microprocessor. It uses interviews and archival documents to discuss the contributions of Fairchild, IBM, and Intel. Bassett makes a particular effort to trace the transfer of ideas between firms, and to tie the dramatically different fortunes of IBM and Intel in turning research into products to their different cultures and internal organizations.

David Caminer, John Aris, Peter Hermon, and Frank Land (eds.), *User Driven Innovation: The World's First Business Computer*. (London, 1996).

A collection of articles, written by the participants, on the origins of the world's first administrative computer (a London tea shop company), its commercialization as a spin-off company (LEO Computers) and its fate.

Martin Campbell-Kelly, *ICL: A Technical and Business History* (New York, 1989).

Covers more than the title suggests, because ICL's origins lay in the punched card era and it represented the merger of almost the entire UK hardware industry. Currently out of print.

James Cortada, *Before the Computer: IBM, Burroughs and Remington Rand and the Industry they Created, 1865-1956* (Princeton, NJ, 1993).

The main history of the pre-computer office machine industry. Charts the shifting fortune of adding machine, typewriter, and bookkeeping machine companies – full of facts and figures. Argument is that computer business evolved out of existing office machine industry.

Robert X. Cringley, *Accidental Empires* (New York, 1996).

A history of microcomputer industry from the mid-1970s to the very early 1990s. Far from academic in its methodology and writing style, but probably more fun to read than any other book ever written on computing. Beneath the folksy exterior and focus on personalities are some vital questions about architecture and technological evolution that historians and economists are yet to really address. And he actually relates the quirks of his subjects to the strategies and fortunes of the businesses they shape.

Michael Hiltzik, *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age* (New York, 1999).

The story of Xerox's Palo Alto Research Center is one of the most famous in the history of computing. During the mid-1970s a fairly small team invented Ethernet and the laser printer, while building the first modern windowing system (inventing icons and greatly improving mouse control and windows) and making huge strides in object oriented programming. Having been handed the key ingredients of a late-1980s workstation about a decade early, Xerox then conspicuously failed to dominate the computer industry, though it did make some money on laser printers. This book is the most thorough account of the PARC story, and while neither exceptionally readable nor exceptionally insightful, it still tells a fascinating tale.

Tracy Kidder, *The Soul of a New Machine* (Boston, MA, 1981). **(Start here)**

Only history because it's old, but still impossible to overlook. This Pulitzer Prize winner is the inside story of the development of a late-1970s minicomputer. Kidder captures the excitement of technological creation, the culture of computer hackers and the gulf between corporate politics and technical striving better than anyone since. It's also a primer on mini-computer design and the computer industry of the era.

Emerson W. Pugh, *Building IBM: Shaping an Industry and its Technologies* (Cambridge, MA, 1994).

A less technical, more general history of IBM and its computers from the creation of the tabulating machine to the crisis of the early 1990s. Based on the IBM archives.

James S. Small, *The Analogue Alternative: The Electronic Analogue Computer in Britain and the USA, 1930-1975* (New York, 2001).

The first comprehensive history of analog computers and computing, an important electronic precursor to modern programmable digital computers. Small shows that the analog computer

industry developed alongside the early digital computer industry and remained vibrant well into the 1960s for scientific, technical and control purposes.

Steven Usselman, "IBM and its Imitators: Organizational Capabilities and the Emergence of the International Computer Industry." *Business History Review* 22/1 (Spring 1993): 1-35. An interesting, analytical article about the institutional development of IBM, its relationship to the broader environment, and a comparison with its competitors in other countries.

John Vardalas, *The Computer Revolution in Canada: Building National Technological Competence* (Cambridge, MA, 2001).

A history of computing in Canada, from the Second World War though to the 1980s. One major theme is the role of military policy in shaping the development of Canadian computing, another is the related push to nurture a self-sufficient computer industry. Essential readings for Canadians, but also of more general interest as a contrast with the US story, particularly because important influences that are often overlooked in the US context become much more apparent here.

Thomas Watson, Jr. and Peter Petre, *Father, Son & Co: My Life at IBM and Beyond* (New York, 1990). **(Start here)**

Watson led IBM to dominance of the computer business during the 1960s, succeeding his father who created the firm. This fascinating biography, produced with a skilled ghostwriter, provides perhaps the most readable introduction to the punched card and mainframe business and provides an insider viewpoint of key moments in its development.

Programming Languages, Software, and the Software Industry

Atsushi Akera, "Voluntarism and the Fruits of Collaboration," *Technology and Culture* 42/4 (October 2001): 710-36.

SHARE was a voluntary organization of large IBM scientific computing installations. It set up a shared library of systems and utility software, and attempted to coordinate the production of the first standard operating system for IBM machines. This has interesting parallels (and differences) with today's open source software movement.

Manfred Broy, and Ernst Denert (eds.), *Software Pioneers: Contributions to Software Engineering*. Berlin: (Berlin, 2002).

Another book based on a conference. This one even includes DVDs of the speeches. In an unusual format, the book is split between often interesting speeches made during the conferences by the pioneers themselves, giving a retrospective view of their accomplishments, and reprints of their crucial original papers. "Software Engineering" is very broadly defined, so as well as the obvious suspects such as Tom DeMarco, Michael Jackson, Fred Brooks and Barry Boehm, the conference included the likes of Peter Chen, David L. Parnas, Niklaus Wirth, Friedrich L. Bauer, and Alan Kay.

Martin Campbell-Kelly. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry* (Cambridge, MA, 2003). **(Start here)**

The first overall history of the software industry, and a very good one too. Software is given a suitably broad interpretation here, to include computer services and timesharing firms in the 1960s and 1970s (though not more recently), as well as packaged applications and systems software. Mainframe and microcomputer software (including games and personal titles) are all covered. However, this is a history of the software *industry* and not of software itself, or for the most part of its usage.

Ulf Hashagen, Reinhard Keil-Slawik, and Arthur L. Norberg, eds. *Mapping the History of Computing: Software Issues*. (New York, 2002).

The proceedings of a workshop at which an eminent group of historians and computer scientists got together to examine different ways of thinking about the history of software (“as science”, “as engineering,” etc.) and to try and come up with a research agenda for the future. Lengthy comments and discussion summaries are included as well as the papers themselves. Many of the papers related to software engineering, in one way or another.

Steven Levy, *Hackers: The Heroes of the Digital Revolution* (Garden City, NY, 1984). **(Start here)**

Levy, a journalist, tells the story of three crucial chapters in the evolution of interactive computing: the obsessive MIT hacker programmers of the 1960s, the personal computer hardware enthusiasts of the 1970s, and the videogame developers of the 1980s. The final section now seems only mildly consequential, but remains quite interesting. The first two sections are extraordinarily vivid and insightful accounts, doing a great job of getting at the motivations and cultures of these influential enthusiasts. His codification of “hacker culture” has been a major influence on today’s free software movement (hacker in this sense comes from MIT traditions, and has little to do with breaking into computers).

Steve Lohr, *Go To: The Story of the Math Majors, Bridge Players, Engineers, Chess Wizards, Maverick Scientists and Iconoclasts—The Programmers Who Created the Software Revolution* (New York, 2001). **(Start here)**

Much better than the cheesy title suggests, this book relies heavily on interviews (Lohr’s and existing oral histories) to retell the origins of the most famous programming systems, from FORTRAN to GNU. Some of these stories have been done to death already (Xerox PARC) but many remain fresh (credit goes for including Visual Basic along with the more theoretically respectable developments). Does a good job of explaining why each one was important and what was novel about it, but doesn’t tell you much about what people actually did with these systems once they were released—in terms of either applications or experiences.

Richard L Wexelblat (ed.), *History of Programming Languages*. New York: (New York, 1981); and

Thomas J Bergin, and Rick G Gibson (eds.), *History of Programming Languages II*. (New York, 1996).

Based on the proceedings of a seminar bringing together the creators of languages such as FORTRAN and BASIC to tell stories about the creation and subsequent evolution of these languages. Each session was devoted to a single language. The strength of this kind of thing is in presenting a fairly polished set of memoirs, and sometimes capturing discussion and disagreement. The second volume, and corresponding seminar, covers newer languages such as C++, LISP, ADA, SMALLTALK and PASCAL. Table of contents available at: <http://www.csis.american.edu/tbergin/pubs/programming.html>.

Communication and Control Systems

Janet Abbate, *Inventing the Internet* (Cambridge, MA, 1999). **(Start here)**

Still the only full-length scholarly history of the emergence of the Internet. Reliable, concise and clearly written. Looks at the origins of packet-switching technology, the creation of ARPANET, the transition to TCP/IP, and the uses made of the system by academic researchers. Although there is a brief discussion of the Web and the commercialization of the Internet, it is something of an afterthought to the focus on the Internet as a scientific network.

Paul Edwards, *The Closed World: Computers and the Politics of Discourse in Cold War America* (Cambridge, MA, 1996). **(Start here)**

Very readable. Includes history of SAGE air defense network, draws parallels with SDI. Shows military origins of real-time computing, networks and other technologies. Edwards looks at broader cultural and political issues, and adopts a cultural studies sensibility in including discussion of science fiction films as well as military systems.

David A. Mindell, *Between Human and Machine: Feedback, Control, and Computing Before Cybernetics* (Baltimore, 2002).

Excellent and complicated treatment of the prehistory of computing, focusing on real-time control and communication technologies during WWII and the preceding couple of decades. These include feedback and communications engineering, naval fire control, guidance technologies, differential analyzers and anti-aircraft systems. Each is situated within a specific tradition of engineering practice, but collectively they lay the groundwork for many of the better known post-1945 developments in computing. More details in my review, at: <http://www.tomandmaria.com/Tom/Writing/MindellReview.pdf>.

Business and Administrative Applications

Atsushi Akera, "Engineers or Managers? The Systems Analysis of Electronic Data Processing in the Federal Bureaucracy." In *Systems, Experts, and Computers: The Systems Approach in Management and Engineering, World War II and After*, ed. Agatha C. Hughes and Thomas P. Hughes, 191-220. (Cambridge, MA, 2000).

A nice case study of early attempts by members of the National Bureau of Standards to use expertise in computer technology and knowledge of the techniques of "systems analysis" to assert broader control over administrative computing across the Federal government.

Jon Agar, *The Government Machine* (Cambridge, MA, 2003).

This history is the first to really explore the use made of the computer and other information technologies by government. The book surveys a very broad range of developments in British government, from Babbage and his famously unbuilt machines, through the much less well known stories of government statistical work, punched card machine use, operations and methods experts, information handling during the World War II, and computerization during the 1950s and 1960s.

Martin Campbell-Kelly, *The Railway Clearing House and Victorian Data Processing* (London, 1994).

A nice case study of how a large scale administrative system functioned long before the introduction of the computer.

Alfred D. Chandler and James W. Cortada, *A Nation Transformed by Information: How Information Has Shaped the United States from Colonial Times to the Present* (New York, 2000).

This hefty anthology is an attempt to explore information as a theme in US History. The focus is on information considered broadly, rather than just the computer, and so the book includes discussion of the U.S. Post Office, radio, telegraphy, and so on. The treatment of computer and precursor technologies is the context of business applications, with Yates and Cortada providing chapters summarizing their main ideas.

Thomas Haigh, "The Chromium-Plated Tabulator: Institutionalizing an Electronic Revolution, 1954-1958," *IEEE Annals of the History of Computing* 23/4 (October-December 2001): 75-104;

and

Thomas Haigh, "Inventing Information Systems: The Systems Men and the Computer, 1950-1968," *Business History Review* 75/1 (Spring 2001): 15-61.

No book has yet been published to tell the overall story of computer use by American businesses, so I will shamelessly recommend two of my own papers. The former examines the first wave of computer use in business administration, including the processes by which the machines were brought and sold, continuities with earlier punched card applications, and the new jobs and departments created around them. It is available online at: <http://www.tomandmaria.com/Tom/Writing/Chromium-PlatedTabulator.pdf>. The latter examines the redefinition, during the 1960s, of the computer as a tool for "information systems" rather than "data processing," and the role of computer manufacturers and experts in administrative methods of popularizing a new ideal of management by computer. It is available online at: <http://www.tomandmaria.com/Tom/Writing/InventingInformationSystems.htm>.

JoAnne Yates, *Control Through Communication: The Rise of System in American Management* (Baltimore, MD, 1989).

This book deals with the late 19th century and the first few years of the 20th, so does not address the computer at all. It is, however, a well-researched and readable history of the introduction and use of earlier administrative technologies such as files, adding machines, typewriters, carbon paper and memoranda. These are the things that, one by one, the computer has replaced. The first part covers the technologies; the second part gives case studies.

JoAnne Yates, "Co-Evolution of Information-processing Technology and Use: Interaction Between the Life Insurance and Tabulating Industries." *Business History Review* 67, no. 1 (Spring 1993): 1-51;

and

JoAnne Yates, "The Structuring of Early Computer Use in Life Insurance." *Journal of Design History* 12:1 (1999): 5-24;

and

JoAnne Yates, "Application Software for Insurance in the 1960s and Early 1970s." *Business and Economic History* 24:1 (Fall 1995): 123-34.

Yates has been working for some time on a book providing a detailed case study of the use of computers and punched card machines in the insurance industry. These papers present parts of that broader project.

History of Scientific Computing

William Aspray (ed.), *Computing Before Computers* (Ames, IA, 1990).

This collection provides a readable introduction to the various computer technologies used for scientific calculation before the spread of the digital electronic computer, including analog computers and punched card machines. It is now out of print, but the full text is online at <http://ed-thelen.org/comp-hist/CBC.html>.

William Aspray, *John von Neumann and the Origins of Modern Computing* (Cambridge, MA, 1990).

Rather than trying to cover the whole of von Neumann's life, Aspray uses an examination of his contributions during the 1940s and 1950s to computer design, numerical analysis, computing theory, and several other areas as a window into the broader history of computing and applied mathematics during this period.

Martin Campbell-Kelly, Mary Croarken, R. Flood, and E. Robson (ed.), *The History of Mathematical Tables: From Sumer to Spreadsheets* (New York, 2003).

Mathematical tables might not sound so thrilling, but they were one of the main items driving the development of scientific computing technology and techniques during the nineteenth and early twentieth centuries. This book, based on a conference but very nicely edited and produced, provides concise, readable and authoritative introductions to a number of topics, including Babbage and his engines, large scale table-making efforts in the UK and in the US, and the production of nautical and astronomical tables. Most of the authors are professionally trained historians.

Mary Croarken, *Early Scientific Computing in Britain* (Oxford, 1990).

A concise history of British computation, focusing on the development of different computing centers and their use of mechanical and electronic aids. It begins in the early twentieth century, moves through the introduction of differential analyzers, and finishes with the World War II and the electronic computers of the immediate post-War era. No such overall history has yet been published on the US story over the same period.

Paul N. Edwards. "The World in a Machine: Origins and Impacts of Early Computerized Global Systems Models." In *Systems, Experts, and Computers : The Systems Approach in Management and Engineering, World War II and After*, ed. Agatha C Hughes and Thomas P Hughes (Cambridge, MA, 2000), 221-53.

Edwards should soon be publishing a book on the use of computers to model the world's atmosphere, climate, and other "dynamics," something initially viewed as vital applications though it proved much harder to do and more limited than expected, eventually providing the big cliché of chaos theory (you know, the one about the butterfly flapping its wing and making a storm). Until the book arrives, you can read this early presentation of some of the material. Edwards has helpfully placed it online at <http://www.si.umich.edu/~pne/PDF/wiam.pdf>.

Herman H. Goldstine, *The Computer from Pascal to von Neumann* (Princeton, NJ, 1972).

Mixes the early history of calculating devices with an insider's perspective on the development of the computers of the 1940s and 1950s.

David Alan Grier, "The Math Tables Project of the Work Projects Administration: The Reluctant Start of the Computing Era." *IEEE Annals of the History of Computing* 20/3 (July-September 1998): 33-50;

and

David Alan Grier, "The Rise and Fall of the Committee on Mathematical Tables and Other Aids to Computation." *IEEE Annals of the History of Computing* 23/2 (April-June 2001): 38-49.

Grier will soon publish a book tracing scientific computing in the United States from the origins of numerical analysis to the beginnings of digital electronic computing. In the meantime, several articles from the same research have been published.

Jennifer S. Light, "When Computers Were Women." *Technology and Culture* 40/3 (July 1999): 455-83.

The initial programmers and operators of the ENIAC, the first usable electronic digital computer, were largely women. In this paper, Light tells their story and examines their disappearance from the official record.

Stephen G. Nash, ed. *A History of Scientific Computing* (New York, 1990).

This edited publication based on a conference (the proceedings of which are available to members in the ACM Digital Library). The book is unobtainable at present (mid-2004), but should soon be appearing online, hosted by the Society for Industrial and Applied Mathematics. It consists mostly of first-person accounts by pioneers in scientific computing and numerical analysis.

Jeffrey R. Yost, *A Bibliographic Guide to Resources in Scientific Computing, 1945-1975* (Westport, CT, 2002).

An annotated list of more than a thousand primary and secondary sources (books, articles and archival papers) related to different kinds of scientific computing. These are grouped according to the main area of science (physical sciences, biological sciences, cognitive science, and medicine). As this is such a huge topic, and this is quite a slim book, it's inevitably far from complete, but it might point to some good first steps for student research projects.

History of Computer Science and Research

William Aspray, and Bernard O. Williams, "Arming American Scientists: NSF and the Provision of Scientific Computing Facilities for Universities, 1950-73." *IEEE Annals of the History of Computing* 16/4 (Winter 1994): 60-74;

and

William Aspray, "Was Early Entry a Competitive Advantage? US Universities That Entered Computing in the 1940s." *IEEE Annals of the History of Computing* 22/3 (July-September 2000): 42-87.

Nobody has ever written a book giving the overall history of computer science as an academic discipline. One avenue would be to focus on the intellectual content of computer science, particularly theoretical computer science, and examine its connection with other disciplines such as mathematics and engineering. Another would be to look more sociologically at the institutional development of journals, departments, and funding sources in the new field. In these two articles, Aspray takes the latter approach and fills in important parts of this history.

Thierry Bardini, *Bootstrapping: Douglas Engelbart, Coevolution, and the Origins of Personal Computing* (Stanford, 2000). **(Start here)**

Despite the occasional theoretical flourish (Bardini teaches communications, not history) this is a clearly written, involving, and solidly researched look at a previously hazy chapter in computer history. It sets Doug Engelbart's invention of the mouse in the broader context of his '60s Californian philosophy, the institutional history of his research group, and the development of computer technology.

Alan Hodges, *Alan Turing: The Enigma of Intelligence* (New York, NY, 1983).

An excellent biography of the founding father of computer theory. Very long, but has drawn wide public interest, partly by making Turing a gay hero, and partly by documenting his wartime work cracking the German Enigma code, an inherently exciting chapter in the history of computing.

Arthur L. Norberg and Judy E. O'Neill. *Transforming Computer Technology: Information Processing for the Pentagon, 1962-1986* (Baltimore, 1996).

A history of ARPA's celebrated work in supporting the development of seminal computer science research during the 1960s and 1970s. The style is dry, but the material is fascinating. Different chapters explore work in computer graphics, artificial intelligence, and networking (ARPA was responsible for creating ARPANET, the precursor to the Internet).

Alex Roland and Philip Shiman, *Strategic Computing: DARPA and the Quest for Machine Intelligence* (Cambridge, MA, 2002).

This volume covers the efforts of DARPA leaders of the 1980s to shackle together an unrelated mass of apparently promising areas of basic research and sell them to Congress as an applied development effort in military applications. As well as being a fascinating case study in the interaction of computer science research with government agencies and military priorities, it also documents some important areas of recent history, particularly the push for a "fifth generation," and attempts to push technologies for natural language recognition, machine vision and expert systems into practical applications.

Donald MacKenzie, *Mechanizing Proof* (Cambridge, MA, 2001).

MacKenzie is an historically minded sociologist of science and technology, who likes to work by getting deep inside the technical discussions of the community he is studying. This book is a series of case studies, all exploring different aspects of the relationship between computer technology and mathematical proof. Some chapters explore use of computers to produce mathematical proofs, such as the Four Color Problem. Most of the book, however, examines attempts to formally prove the correctness of hardware and software, an important research area in theoretical computer science from the 1960s onward, and one attempted for some real-world military systems.

Mitch Waldrop, *The Dream Machine: JCR Licklider and the Revolution that Made Computing Personal* (New York, 2001). **(Start here)**

A big sprawling book. Waldrop uses Licklider, an experimental psychologist with a vision of man-machine interaction who controlled DARPA's computing research funding during the early 1960s, as a vehicle to structure a much broader story about the development of interactive computing. He starts with the World War II, and takes in cybernetics, artificial intelligence, timesharing operating systems, the Arpanet, and the famous research at Xerox PARC. The style is readable and journalistic, so this would be a nice introduction to the topic. The paperback version has a truly dreadful cover, but don't let that put you off.

Michael S. Mahoney. "Software as Science—Science as Software." In *Mapping the History of Computing: Software Issues*, ed. Ulf Hashagen, Reinhard Keil-Slawik and Arthur L. Norberg, 25-48 (New York, 2002).

Among other things, this complex paper is a discussion of the emergence of theoretical computer science from various obscure areas of mathematics. The text is available online at <http://www.princeton.edu/~mike/softsci.htm>.

National Research Council. *Funding A Revolution: Government Support for Computing Research* (Washington, DC, 1999).

A blue ribbon science panel produced a study showing the influence of government support on the development of computer science, hardware and software. The full text is on-line at <http://www.nap.edu/readingroom/books/far/contents.html> .

Other Kinds of Resources

Articles about History of Computing

Below is a sampling of provocative articles about the current state of the history of computing and its future. Most have been placed on-line by their authors. These all constitute thoughtful and fairly recent reviews of some important works in the field

Michael S. Mahoney, "The Histories of Computing(s)", a lecture in the series *Digital Scholarship, Digital Culture*, at the Centre for Computing in the Humanities, King's College, London, 18 March 2004 (to appear); and

Michael S. Mahoney, "Issues in the History of Computing," in Thomas J. Bergin and Rick G. Gibson ed., *History of Programming Languages II* (New York, 1996), 772-81; and

Michael S. Mahoney, "The History of Computing in the History of Technology", *Annals of the History of Computing* 10 (1988): 113-25.

At frequent intervals, Mahoney delivers a speech or writes an article directed at a different audience summing up his view of the current state of the history of computing and its possible avenues of development. These papers, among others, are available from <http://www.princeton.edu/~mike/computing.html>.

Nathan Ensmenger, "Power to the People: Toward a Social History of Computing." *IEEE Annals of the History of Computing* 25:1 (January-March 2004): 93-95; and

Paul N. Edwards, "Making History: New Directions in Computer Historiography." *IEEE Annals of the History of Computing* 23:1 (January-March 2001): 85-87.

Two short articles containing some thoughts on the future of the field. Edwards offers a number of ideas, while Ensmenger on the desirability of using the tools of social history to study the history of computing. Every issue of *Annals of the History of Computing* includes a "Think Piece" editorial at the back, where someone involved with the field contributes a short article presenting their suggestions on how to think about the field. Unfortunately you need a subscription to read these, though Edwards put his online at <http://www.si.umich.edu/~pne/PDF/makinghistory.pdf>.

Magazines and Journals with History of Computing Content

Technology and Culture.

(<http://www.shot.jhu.edu>)

Produced by the Society for the History of Technology. (recent volumes on-line in Project Muse). SHOT is the leading organization for historians of technology.

Business History Review

(<http://www.hbs.edu/bhr/>)

Longest established journal of its kind, has published several articles on the history of computing over the past few years. Produced by the Harvard Business School (recent volumes in Proquest and Academic Index).

Isis

The leading history of science journal, produced by the History of Science Society. Hardly ever publishes computer-related articles, but we can hope.

American Heritage of Invention and Technology

(www.americanheritage.com/)

Readable, often well researched, aimed at public, frequently includes articles on computer and electronic technologies.

In addition, there are many professional journals in related fields, including IEEE, SIAM and ACM publications, and the Journal of ASIS&T, that occasionally publish historical articles.

Personal Home Pages of Historians of Computing

These generally include links to the text of publications, syllabi, and other relevant material.

Jon Agar –http://www.man.ac.uk/Science_Engineering/CHSTM/people/agar.htm

Atsushi Akera –<http://www.rpi.edu/~akera/>

Thomas J. Bergin –<http://www.csis.american.edu/tbergin/>

Michael Buckland –<http://www.sims.berkeley.edu/~buckland/>

Martin Campbell-Kelly –<http://www.dcs.warwick.ac.uk/~mck/>

Greg Downey –<http://www.journalism.wisc.edu/~downey/>

Paul Edwards –<http://www.si.umich.edu/~pne/>

Nathan Ensmenger –<http://www.sas.upenn.edu/~nathanen>

Thomas Haigh (author) –<http://www.tomandmaria.com/tom>

John A.N. (JAN) Lee –<http://ei.cs.vt.edu/~janlee/Janlee.html>

Timothy Lenoir –<http://www.stanford.edu/dept/HPS/TimLenoir/>

Michael Mahoney –<http://www.princeton.edu/~mike/>

JoAnne Yates –<http://ccs.mit.edu/yates.html>

Thomas Haigh is an assistant professor in the School of Information Studies of the University of Wisconsin, Milwaukee. Haigh holds a Ph.D. from the University of Pennsylvania, and has published articles on the history of the software industry, the origins of the database management system, the historical relationship of computing to corporate management, and the creation of the data processing department. His current research projects include the social history of the personal computer and the origins of open source software in the scientific computing field. He serves as Biographies Editor of IEEE Annals of the History of Computing, and is a member of the Committee on ACM History. *Author's address:* SOIS, Bolton Hall, Room 568, P.O. Box 413, Milwaukee, WI 53201 USA. Website: www.tomandmaria.com/tom. Email: thaigh@acm.org.

27. Resources for a Course on the History of Computing

Thomas J. Bergin
American University in Washington

Jack Hyman
SRA International

Abstract

This paper describes the Computing History Museum's website which was created to support faculty teaching: 1) courses in the history of computing, or 2) courses in which material on the history of computing would be appropriate, such as an Introduction to Computing. This paper is a companion to the "History of Computing for Computer Science and Information Systems Students" and "Enhancing Learning Using the History of Computing: Opportunities to Use History in an Introductory Computers and Information Course," presented in chapters 10 and 11 of this volume. Faculty interested in adding material on the history of computing should read either of these papers before reading this one, which specifically focuses on the various resources available at the Computing History Museum website: www.computinghistorymuseum.org (shown in Figure 1 below).²⁰¹

In the fall of 1998, Tim Bergin received a grant from the Alfred E. Sloan Foundation as part of their *Science and Technology in the Making (STIM)* project. The purpose of the *STIM* project was to capture the history of various technologies that Sloan believed would be ignored by historians because science and technology were expanding so rapidly. The *STIM* projects were experiments to see if people's experiences with various new technologies could be documented using web-based tools.

Our research team created a Web site: (URL: <http://computinghistorymuseum.american.edu/>) to capture the histories of: 1) programming languages, and 2) software engineering. In addition, the website contained photographs of artifacts in the *Computing History Museum*, which we hoped would bring people to the site and, thus, further the project goals.

In December 2000, as part of a request for a no-cost extension, we requested approval to enlarge the site by adding two additional sections: the first to support *teaching the history of computing* and the second to support the Editorial Board of the *IEEE Annals of the History of Computing*.²⁰² Dr. Bergin had been teaching a course in the history of computing since 1988, and wanted to make the syllabus and lecture notes (in the form of PowerPoint slides) available to others in the hopes of helping them teach this important subject. A discussion of this course is contained in the companion paper, "History of Computing for Computer Science and Information Systems Students" (see chapter 10, this volume).

²⁰¹ The Computing History Museum occupied two rooms in the foyer of Clark Hall on the American University campus. Unfortunately, the Museum no longer exists. After Dr. Bergin retired in 2002, the University renovated Clark Hall for the School of International Service. A virtual tour of artifacts is still available on the CHM website and may be useful to faculty and students interested in the history of computing.

²⁰² The *IEEE Annals of the History of Computing* is the journal of record for this subject; Dr. Bergin served as the Editor-in-Chief from 1999 to 2003 and as Assistant Editor-in-Chief from 1996 to 1999.

Figure 1. Computing History Museum Home Page



CSIS 550: History of Computing

CSIS 550 was a one-semester course that covered the history of computing from the abacus to the present. It was an elective course in the Department of Computer Science and Information Systems (CSIS) at American University. At the time, the department had approximately 180 undergraduates (about 60 computer science majors and 120 information systems majors) and about 200 graduate students. About eighty percent of the students who took CSIS 550 were information systems majors; computer science students also took the course, as did occasional students from other departments (especially from the business school). Faculty interested in teaching a course in the history of computing should read the companion piece in this volume first (see chapter 10) and then use this paper to explore the website and how it can help them in preparing for their lectures. Figure 2 opens the section on teaching the history of computing.

Although the purpose of putting this material on the Web is to support faculty who intend to teach the history of computing, *it does not substitute for knowledge of the subject area*. Gaining a thorough knowledge of computing history takes considerable time and effort; this website should facilitate the "packaging" of relevant historical material for classroom use, that is, through the availability of a set of lectures that can be modified by users to meet their teaching needs.

Figure 2. Teaching the History of Computing Page



The site contains (in order): 1) the syllabus for CSIS 550, The History of Computing; 2) PowerPoint® slides for 15 lectures; 3) class materials; 4) sample student projects; 5) a discussion page; 6) a bibliography; and 7) other resources to aid in course preparation (information about videos and museums). In addition to the course lectures, there is an “Overview of the History of Computing,” suitable for a lecture of 90 to 120 minutes in an introductory computing course or as a module in other related courses.

As shown in Figure 3, the lectures can be viewed or downloaded. In this way, faculty can look at a lecture to see if it meets their needs, and then download and modify it for content and personal style. In the past two years, a number of instructors from other universities have requested permission to use the slides in their classes. Although permission is not required, we do ask that they use “normal academic courtesies” with respect to citing the Computer History Museum as the source of the materials, and that they provide us with feedback so that we might correct mistakes or add materials in the future.

Figure 3. Lectures in the History of Computing

American University Computing History Museum
Home - Museum - Teaching the history of computing - Class project - IEEE Annals

Lectures in the History of Computing

This page contains Professor Tim Bergin's lectures from the class CSIS 550, History of Computing. We are providing two formats of lecture slides - MS PowerPoint for download and HTML to view. You can always come back from the slides to this page by clicking on the Back button on the browser window.

Navigation Menu:
Introduction
Syllabus
Lectures
Class Materials
Student Projects
Discussion
Bibliography
Resources

Lecture 1: Overview of the History of Computing

- History of Computing Overview ([View](#) | [Download](#))


Lecture 1: Numeration

- Numeration ([View](#) | [Download](#))


We think it important to make two observations at this point: first, the reader should log on to the Computing History Museum website (URL: <http://computinghistorymuseum.american.edu/>) in order to fully follow this discussion. In this way, you can view an entire lecture and gain some sense of how it might fit your requirements. A second observation is equally practical: a valuable feature of using PowerPoint®s that the lecture slides can be printed (in black and white) as student handouts. Figure 4 is a sample handout from the lecture on Charles Babbage.

Figure 4. Sample of Class Handout

Charles Babbage (1791-1871)



Never to be completed



- December 1830, a dispute with his chief engineer, Joseph Clement, over control of the project, ends work on the difference engine
- Clement is allowed to keep all tools and drawings by English law


Charles Babbage (1791-1871)

- Born: December 26, 1791
- son of Benjamin Babbage a London banker [part of the emerging middle class: property, education, wealth, and status]
- Trinity College, Cambridge [MA, 1817] with John Herschel and George Peacock, produced a translation of LaCroix's calculus 1802

Importance of the Difference Engine

- 1. First attempt to devise a computing machine that was automatic in action and well adapted, by its printing mechanism, to a mathematical task of considerable importance.
- 2. An example of government subsidization of innovation and technology development
- 3. Spin off to the machine-tool "industry"

A vision of calculating by steam!



My friend Herschel, calling upon me, brought with him the calculations of the computers, and we commenced the tedious process of verification. After a time many discrepancies occurred, and all one said these discrepancies were so numerous that I exclaimed 'I wish to God these calculations had been executed by steam.' 1822

Science Museum's Reconstruction

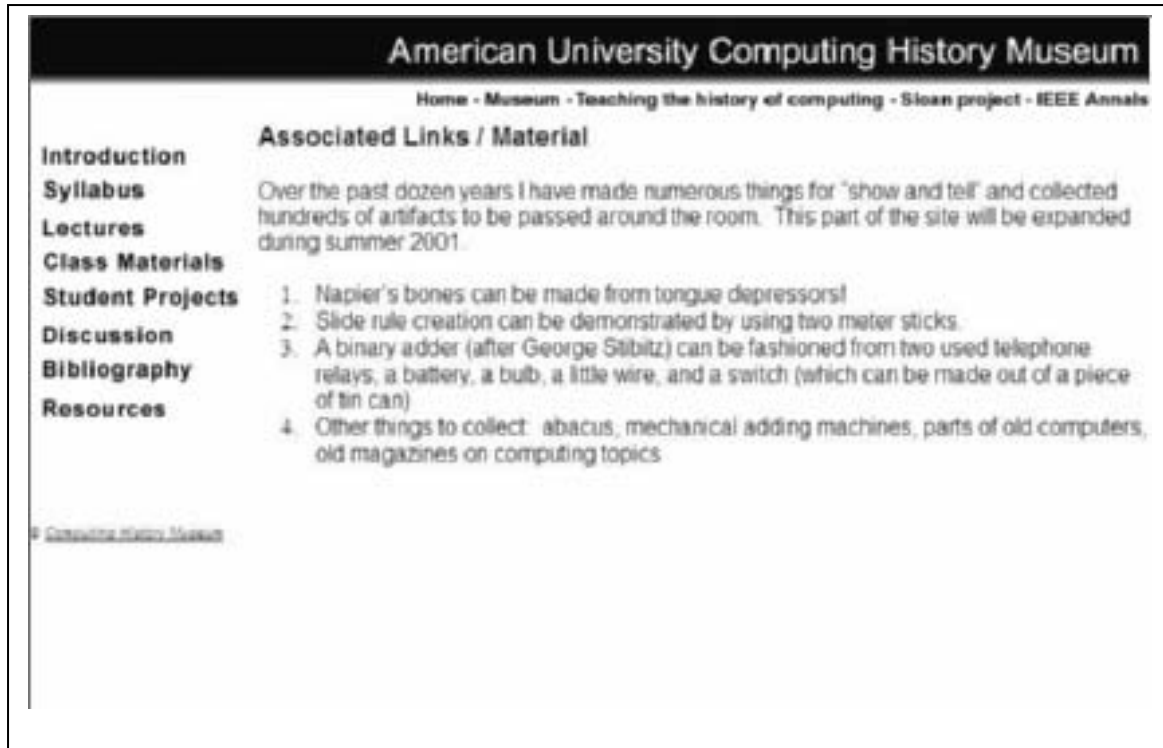
- Difference Engine Number 2 (1847 to 1849) constructed according to Babbage's original drawings (minor modifications)
- 1991 Bicentenary Celebration
- 4,000 parts
- 7 feet high, 11 feet long, 18 inches deep
- 500,000 pounds

In addition to the syllabus and lecture slides, the site contains other materials of use to the potential instructor.

Class Materials

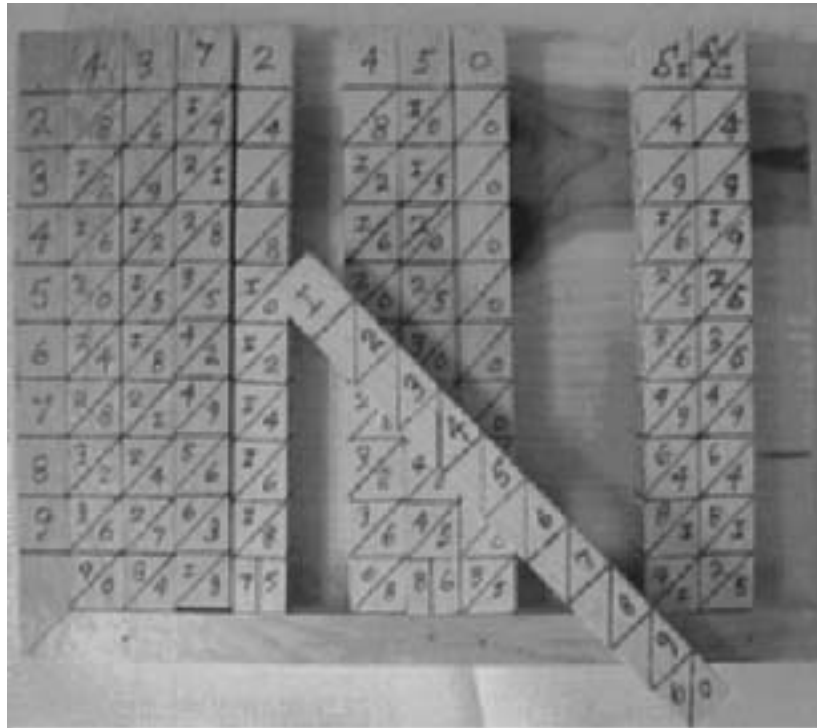
The Class Materials page contains a few hints on artifacts that can be created for the classroom as shown in Figure 5.

Figure 5. Associated Links/Material Page



One of the first artifacts that Dr. Bergin made was a set of Napier's "bones." He used some scrap balsa wood from a project. Each "bone" was an inch wide so the numbers could be read from the back of the room. It was a big hit! To read about things is good, but to see an artifact being used in the classroom is better. In this case, the replica wasn't close to the original, but it allowed students to see for themselves how they worked. An even better strategy is to actually put the artifact in the student's hands and let them try to use it. When the Museum was being constructed, a carpenter from the university's building department helped to create a better replica, that is, one that was physically larger (so the numbers can be seen in a classroom), but where the "bones" were square—and thus display four sets of partial products—as in Napier's originals. When he finished cutting and sanding the "bones" all that had to be done was to draw the lines and add the partial products. The final product is shown in Figure 6.

Figure 6. Napier's Bones (home made)



The creation of artifacts can also occur as student projects. One adult student needed one credit hour to graduate. Since he worked for the local telephone company, he was able to find some old telephone relays and build a replica of the binary adder created by George Stibitz in 1937.²⁰³ Indeed, the Computing History Museum's virtual tour was created as a student project, as were other parts of the site.

The artifacts contained in the Museum were used throughout the class to enliven discussions and to put some "meat on the bones" of history. It is easy to find slide rules, old calculators and other interesting objects if one tries.²⁰⁴ Another interesting demonstration is to make a slide rule. This is easily done using meter sticks, some colored stick-on circles, and a list of the logarithms of 1 to 10.

Recently, Dr. Bergin has been working as a volunteer in elementary schools, and made a "table abacus" from a piece of foam core (30" by 40"). The lines were made from Velcro® and the counters are 1 ½" wooden disks, purchased at a local craft store and painted black to contrast with the white "abacus." The bottom line is that with a little imagination, ingenuity, and

²⁰³ This was later referred to as the "Kitchen computer or K-1" because Stibitz, a mathematician at Bell Telephone Laboratories, built it in his kitchen from components that he had taken from the shop scrap heap.

²⁰⁴ An article about the Museum and how the artifacts were collected appeared on Yesterday's Office (<http://www.yesterdaysoffice.com/index.cfm?fuseaction=ShowArticle&articleid=36%20>). There is a link to the Yesterday's Office site on the Museum page, as well as a longer version of the Museum's history.

time, instructors can make a number of objects that will make the study of computing history come alive for students.

Student Projects

The Student Projects pages contain sample *biography*, *period*, and *graduate research* papers in *Adobe Acrobat*.²⁰⁵ The *biography paper* examines the life and contributions of a significant person in the evolution of computing. The website contains a list references to books that have short biographies of significant people in the history of computing. Students can use these books to get their research started, since they contain short biographies, lists of publications, and references to additional sources.

Since it is impossible to cover the period from the early 1950s to the present with any depth, students are required to examine *Datamation* and *Computerworld* for an assigned year, and write a *period paper* describing the most important computer news for that year.²⁰⁵ Students are encouraged to scan the tables of contents of the issues for that year and look for important hardware, software, or industry topics, including advertising. The period paper is presented to the class. Most students introduce their year by discussing what was going on in the world generally (from *Time*, *Newsweek*, *Business Week*, etc.).²⁰⁶

Graduate students are required to write a *research paper*, which takes an element of current computing and traces it to its origins. Contemporary computer users take CD-ROMS for granted, but *who* did the early research into external storage systems and *what* were they researching at the time? *Which* companies built the first external storage systems for commercial use, and *how* did they work? *What* was the capacity of the first magnetic tape system? *What* were the major milestones leading up to the current form of this technology?

We might think that there is no need to teach the history of computing to computer science or information systems students, especially those studying at the graduate level. However, let us share two anecdotes that prove this assumption wrong. As part of the class on Jacquard and Hollerith, Bergin discussed the invention of the punched card and the evolution of the IBM Electric Accounting Machine (EAM). After the lecture, we would take the students to the Museum to see how an IBM 029 Keypunch punched cards. One evening, a graduate student asked: "Hey, Professor! What PC did this work with?" He might have *meant to say* "computer" but he didn't—he said "PC!" The second anecdote concerns an instructor in our department who had an MS in computer science from a good program. Before a faculty meeting, he asked: "How do you say that guy's name...*Atta...Atta....*?" Bergin replied: "I guess you mean *Atanasoff* as in John Vincent Atanasoff." Here is someone with both bachelors and masters degrees in computer science who couldn't pronounce the name of someone who built a prototype special-purpose calculating device in the late 1930s.²⁰⁷ Suffice it to say that we all have a responsibility to assure that our students are not ignorant of the history of their chosen discipline.

²⁰⁵ Although there were other general interest periodicals during these years, restricting student research to *Datamation* and *Computerworld* provides some continuity between student presentations and makes it easier for the instructor to master the material. *Datamation* has published issues to commemorate milestones in their publishing history, and which contain historical timelines of events: 25th Anniversary in September 1982, "Reaching 30 Years" in the September 15, 1987 issue, and "40 Years of Computing" in the March 15, 1991 issue.

²⁰⁶ Student presentations use PowerPoint or other presentation software and are planned for eight to ten minutes, including questions.

²⁰⁷ After the meeting, we went to lunch and Bergin explained what Atanasoff did, how he did it, the controversy surrounding his research, and the Honeywell/UNIVAC patent suit.

Discussion

The discussion page was created as a place where interested persons could communicate with the Sloan team about the website and teaching the history of computing. Dr. Bergin prepared an opening commentary but very few replies or comments were received. In retrospect, the use of email is probably more efficient than having to check to see if someone sent a note to the website. Thus, interested readers are strongly encouraged to contact the authors directly. Please direct questions about the History of Computing course to Tim Bergin and technical questions about the website to Jack Hyman.²⁰⁸

Bibliography

The bibliography pages contain references to books grouped according to the following categories: General Reference, History of the Computer, Pre-computing, Micro-computing, Hardware, Programming Languages, the Internet, Biographies, Corporate Biographies, and Primary Sources, as shown in Figure 7.

Figure 7. Bibliography Page

The screenshot shows the website for the American University Computing History Museum. The header includes the museum's name and a navigation menu with links to Home, Museum, Teaching the history of computing, Sloan project, and IEEE Annals. The main content area is divided into two columns. The left column contains a vertical list of navigation links: Introduction, Syllabus, Lectures, Class Materials, Student Projects, Discussion, Bibliography, and Resources. The right column is titled 'Bibliography' and contains the following text: 'After almost twelve years of teaching the course, I want to emphasize that there are a number of textbooks that can be used to support a course in the history of computing, depending on the scope of the material to be covered. The section below contains an annotated bibliography which is categorized by type of literature.' Below this text, it states: 'In addition, the site contains references to additional resources such as videos, periodical literature, web sites and the IEEE Annals of the History of Computing.' A final sentence reads: 'Faculty are encouraged to acquire books on the following subjects.' This is followed by a bulleted list of categories: Textbooks, General References, History of the Computer, Precomputing, Microcomputing, Hardware, Programming Languages, The Internet, Biographies, Corporate Biographies, and Primary Sources.

²⁰⁸ Our contact information is contained in the biographies at the end of this chapter.

For this section, one of the graduate students working on the project created a short summary of each book and included a thumbnail of the book's cover (for some of the books in each section). This can save considerable time in trying to find material about specific topics, but does not take the place of research tools such as J.A.N. Lee's *Computer Pioneers* or James Cortada's *A Bibliographic Guide to the History of Computing, Computers, and the Information Processing Industry*. Another important source of information is *The Timetables of Technology: A Chronology of the Most Important People and Events in the History of Technology*, by Bunch and Hellemans. These are all referenced in "General References," as shown in Figure 8.

Figure 8. Bibliography –General References

The screenshot shows the website for the American University Computing History Museum. The header includes the museum's name and navigation links: Home, Museum, Teaching the history of computing, Sisson project, and IEEE Annals. A sidebar on the left lists menu items: Introduction, Syllabus, Lectures, Class Materials, Student Projects, Discussion, Bibliography, and Resources. The main content area is titled "Bibliography - General References" and lists three books:

- Textbooks | History of the Computer | Precomputing | Microcomputing**
- Classrooms | Programming Languages | The Internet | Disruptions | Corporate Disasters | Primary Sources**
- Bryon Bunch and Alexander Hellemans, *The Timetables of Technology: A Chronology of the Most Important People and Events in the History of Technology*, New York: Simon & Schuster, 1993. ISBN: 0671769189. Contains over 5000 entries. Provides a chronological timeline of the significant events in the history of technology.** (Accompanied by a thumbnail of the book cover)
- Bryon Bunch and Alexander Hellemans, *The Timetables of Science: A Chronology of the Most Important People and Events in the History of Science*, New York: Simon & Schuster, 1988. ISBN: 0671621300. A valuable reference on the history of science. The information is in the timetables, a chronological, subject-by-subject chart.** (Accompanied by a thumbnail of the book cover)
- James W. Cortada, *A Bibliographical Guide to the history*** (Accompanied by a small thumbnail)

Resources

The Resource pages contain references to museums, videos and websites useful to faculty teaching the history of computing. There are numerous, high-quality websites today that

focus on the history of computing.²⁰⁹ The page devoted to Museums categorizes museums by “Top Ten, United States, Australia, Brazil, England, France, Germany, Japan, Portugal, Russia, and Sweden,” as shown in Figure 9.²¹⁰

Figure 9. Resources –Computing History Museum > Top Ten Page



In addition, the Resource page lists *Videos* and *Web sites* that can be used to support the teaching of the history of computing, either viewed as part of a class or as an assignment. Showing short excerpts from Volume 1 (Great Brains) of “The Machine That Changed the World” was an effective way to get students to realize how big the early computers were. This experience was enhanced by bringing in modules from a UNIVAC I, which was in the Museum collection. Equally effective was having students watch the entire tape and write a summary of what they learned as an extra-credit exercise. Another, very popular, extra-credit exercise was having students visit the “Information Age” exhibit at the National Museum of American History, Smithsonian Institution, a short Metro ride from the American University campus.

²⁰⁹ These pages were completed in spring 2002 and do not reflect newer websites and displays. The use of a search engine will provide numerous sites to be explored by both faculty and students.

²¹⁰ Since this categorization was done in 2001, there are additional websites that should be listed on this page.

Computer History Museum Virtual Tour

A final resource for instructors and students is offered by the Virtual Tour of the exhibits. As mentioned earlier, this page was created by three students who were taking the Information Systems Workshop Class. This was a semester-long project which required them to photograph (numerous times) the artifacts in the Museum and create a Web page to display the pictures.²¹¹ Two examples of artifacts in the Museum are shown in Figures 10 and 11.

Figure 10. Chinese Swan Pan (Abacus)



Figure 11. Schickard Calculator (Re-creation)



²¹¹ The Web pages shown are not the original pages created by the students, but a new edition done by one of the undergraduate assistants on the Sloan project. Please note that the “History of Computing Online” page, which was created in 1989 by two IS graduate students, Eric Ashman and Seth Gordon, has been corrupted and hopefully will be restored in the near future.

Conclusion

Creating and teaching CSIS 550, History of Computing was the highlight of Dr. Bergin's teaching career. It allowed him to merge teaching and research in ways that improved both activities. The Sloan grant allowed him to recruit a team of undergraduate and graduate students and create the website described in this paper. One of these students was Jack Hyman, who recently finished his masters in education at George Washington University. We are applying for a grant that will give us the resources to update the site and add additional materials. In the meantime, we hope that this site will assist faculty and students in learning and appreciating the history of computing.

Thomas J. (Tim) Bergin is Emeritus Professor of Computer Science at the American University in Washington, D.C. Dr. Bergin entered the computing field in 1966, as a "digital computer systems analyst (trainee)" with the U.S. Veterans Administration. In 1980, he received an Inter-governmental Personnel Act Fellowship as a Visiting Professor in American University's Center for Technology and Administration. In 1982, he joined AU as an Assistant Professor and Director of the Quantitative Teaching and Research Laboratory, and managed the introduction of personal computers to support faculty and student research. He has authored or edited 4 books: *History of Programming Languages* (ACM Press, 1996) and *Fifty Years of Army Computing: From ENIAC to MSRC* (Army Research Laboratory Special Report-93, September 2000), *A Microcomputer-Based Primer on Structural Behavior* (with James Lefter, Prentice-Hall, 1986) and *Computer-Aided Software Engineering* (Idea Group Publishing, 1993) In addition to his teaching duties, Dr. Bergin created the Computing History Museum at American University, which was renovated out of existence upon his retirement in 2002. To see some of the artifacts mentioned in this article you can visit the Virtual Computing History Museum (<http://computinghistorymuseum.american.edu/>). A story about the creation of the Museum was also featured in an article in *Yesterday's Office*.²¹² Dr. Bergin also served as the Assistant Editor-in-Chief of the *IEEE Annals of the History of Computing* from 1995 to 1999 and as Editor-in-Chief from 1999 to 2003. Dr. Bergin also served as historian to the ACM for their 50th anniversary celebration. *Author's address:* 217 Pewter Lane, Silver Spring, MD 20905 USA. Email: tbergin@american.edu.

Jack A. Hyman is a software engineer with SRA International Inc. He has been a member of the Project Management Institute, International Webmasters Association, IEEE, ACM, and The e-Learning Guild. He holds an M.A. in Education & Human Development, Educational Technology Leadership from The George Washington University and a B.S. in Computer Information Systems from The American University. His areas of interests include: usability engineering, content and knowledge management, computing history, and information design. Jack joined the Sloan Research Team as a freshman (in 1998) and played a critical role in the evolution of the site since that time, serving as webmaster since 2001. As a senior research project, Jack created a PowerPoint presentation based on Mitch Waldrop's *The Dream Machine*. *Author's address:* 1400 South Joyce Street, Apt 1609, Arlington, VA 22202 USA. Email: jahyman@comcast.net.

²¹² <http://www.yesterdaysoffice.com/index.cfm?fuseaction=ShowArticle&articleid=36>

Appendix 1. List of Additional Workshop Participants

Amherst College August 6-7, 2001

James Tomayko (Carnegie Mellon University)
Gerard Alberts (Nijmegen University)

University of Minnesota April 26-28, 2002

Steve Russ (University of Warwick)
Gordon Davies (Open University)
Arthur Norberg (University of Minnesota)
Elisabeth Kaplan (University of Minnesota)
Michael Williams (University of Calgary)
Joyce Currie Little (Towson University)

Notes

Notes

Notes



1100 17th Street, NW, Suite 507
Washington, DC 20036-4632

E-mail: info@cra.org

Tel: 202-234-2111

Fax: 202-667-1066

URL: <http://www.cra.org>