History in the Computer Science Curriculum

John A. N. Lee
Professor of Computer Science

Department of Computer Science, Virginia Tech
Blacksburg VA 24601-0106, U.S.A.

Prepared for the 6th IFIP World Conference Computers in Education
Birmingham, England
July 24-29, 1995

Abstract

IFIP Working Group 9.7 (History of Computing) is charged with not only encouraging the preservation of computer artifacts, the recording of the memoirs of pioneers, and the analysis of the downstream impact of computer innovations, but also on the development of educational modules on the history of computing. This paper presents an initial report on the study of the history of computing and informatics and preliminary proposals for the inclusion of aspects of the history of computing and informatics in the curriculum of university and college students.

CR Categories: K.2. History of Computing, K.3. Computers and Education
Keywords: Curriculum, Case studies.

INTRODUCTION

The impact of the information revolution on our society and our industry is immense. In our increasing desire to control our own destinies, we seek to understand not only our contemporary technology, but also to recognize the past trends that will predict the future. Looking backward to discover parallels and analogies can provide the basis for discovering the standards by which we judge the viability and potential of a current or proposed activity. Thus it is important that we consider the place of historical study in the framework of the curriculum of students of computer science and informatics.

The study of the history of computing is now at the point where it is a respected element of the study of the history of technology; our early researchers and scholars have been joined by several eminent historians, and the field is producing young scholars who bring further prestige to the field. For the first time the curriculum for computer science, developed by the Association for Computing Machinery (ACM) and IEEE Computer Society Joint Task Force in 1991, included explicit educational modules related to history in four[1] areas:

- Artificial Intelligence,
- Operating Systems,
- Programming Languages, and
- Social, Ethical and Professional Issues

Unfortunately there is a stigma that often accompanies the decision of scholars to study history after a career in science -- a suspicion that they are becoming senile! This is the same stigma that accompanies the movement of a faculty member to the study of the educational needs of computer science, or to studies of the social impact of computing, concerns for the ethical issues in the field, or writing as part of the

[1] History was NOT included in relation to Algorithms and Data Structures, (computer) Architecture, Database and Information Retrieval, Human-Computer Communication, Numerical and Symbolic Computation, and Software Methodology and Engineering. Some of these are quite surprising; the history of Computer Architecture, for example, has been well researched and would provide a series of case studies which would strongly enhance this module.

curriculum. Yet the study of history is as intense as that of the study of the sciences and many of the techniques of searching and researching, deduction and derivation are applicable to many scholarly, scientific activities. The difference, if any, is that the historian needs a bigger desk for the papers and more file space to store them! On the other hand, the techniques of historical research differ little from the techniques used in quality management reviews and software history analyses.

Toynbee's cliché "Those who forget the lessons of history are doomed to repeat it" is an ambiguous concept. On the one hand, there are some lessons that need to be resurrected, repeated, and extended. There are others that, having been forgotten, ignored or put aside, are now in danger of being repeated. We and our students need to know enough about our history not only to protect ourselves against it, but also to learn from it.

THE HISTORY OF COMPUTING

To those of us who got involved in researching our history in the 1970's, the initial impetus was both curiosity and the expectation that what we were to do would be pleasurable. Our naiveté led us to believe that our data collection and preservation activities constituted both the beginning and the end of the process of historical scholarship. It was (and still is) a pleasure to be able to talk to the pioneers and learn of their accomplishments, to have the means to preserve their stories and recognize their achievements. To many of our colleagues history is only the study of an irrelevant past, with no redeeming modern value -- a subject without useful scholarship. In fact some pioneers themselves chose not to participate in preserving their stories claiming that they were more interested in the future than the past, and that science progresses to overcome its past. Even an eminent historian of technology commented the "Computer history is too recent to forget".

Contemporary history is the activity that allows us to look behind us and to recognize, in their lifetime, our pioneers and their contributions. At the same time the delay between the instant of the innovation and the date of review allows us to view history from a reasonable distance -- to increase our objectivity, to improve our ability to compare against parallel developments, and to recognize the downstream impact on today's technology.

THE HISTORY OF FAILURE

John Backus, creator of Fortran and functional programming said in his acceptance speech on receiving the 1994 Charles Stark Draper Award:

> *In science and all creative work we fail -- again and again. For each successful idea, we usually have dozens of others that don't work, no matter how hard we struggle. But in failing we learn a lot.*

Computer science[2] is somewhat unique in that thirty years ago we could start from first principles and give the student everything there was to know about computing. We could show students the best route to a solution for a problem; there were very few alternative paths and the wrong paths were quickly obvious. For anyone who has been in the field for more than a few years, the lessons learned are the lessons, not of documentation, but of experience. Experience, that is, of having made mistakes, having made the wrong decisions, and occasionally of having achieved the hoped for goal. James Horning said of the relationship between experience and mistakes:

Good Judgment Comes from Experience
Experience Comes from Bad Judgment

Yet few of our courses in a university curriculum use examples of failures as part of their learning system. Students make mistakes themselves truly, but the failures of the past are important learning tools. Failure may teach us as much as success but it takes a particular style of learning to appreciate and use that information appropriately. It is

---

[2] It is, of course, an example of Whiggism to talk about Computer Science in 1960 - the concept may have been there, but not the term!

tantamount to the problems solved and the theorems proved by the method of "contradiction".

However, much of our history is the history of successes; our technical journals are commonly repositories of reports of successful ventures, of the concepts that appear to have wide applicability to practice, and to projects that have been determined by peer review to promise the future. In fact, little of our literature or open records contains information on the motivation, the innovation, and the decision making that led to commercial products, to new methodologies, and significant research results. Personal interactions drive people; people make decisions; people make up organizations; but in the literature it often appears that organizations somehow make decisions. Read almost any copy of the current research literature, and one must conclude that ideas appear unsummoned, only one train of thought led to enhancements, and the whole development of the topic proceeded without any side trips or dead ends. We know from our own experiences that this is patently not the case for anything more than the most trivial experiment. There is a possibility that we can look back some time in the future and evaluate the liveliness of these reports. There are exceptions -- Leveson and Clark (1993) for example, and Hendry (1989). Both of these reports look at the reasons for failure; one in the case of a malfunctioning, computer-driven radiation therapy system, and the other in the governmental attempt to control and manipulate intellectual property in computing. The report by Leveson and Clark (1993) dealt with the accidents related to Therac-25 radiation therapy system. This investigation, while dealing only with activities within the past 15 years, is typical of the kind of historical research which has positive downstream impact. Leveson and Clark's analysis of the software errors provides a number of lessons in software development which reinforce the many lessons which have been preached in software engineering classes for a decade -- unfortunately the decade after the installation of the Therac-25 equipment. Undoubtedly the lessons learned from such errors (and the resulting human suffering) is much stronger than those of positivism.

A recent textbook advertisement by a leading publisher of computer science books proclaimed that this new textbook for beginning students identified the "firsts" in the history of computing as one of its major advantages. Firsts are not always important, though there is obviously a common desire to be associated with firsts; we might reword the assertion that "Victory has a hundred fathers but defeat is an orphan"[3] to read "Firsts have a hundred fathers but seconds are orphans". But in fact though many may lay claims to firsts, firsts are often a matter of definition rather than chronology, for almost everything we do can be considered a "first" for some minor distinction -- such as the location of the event, the first use of some element, and so on. Several firsts *can* occur almost simultaneously, each emerging from a primordial soup of technology which contains all the ingredients for the development of a new innovation. Unique firsts do have a place in the identification of the owners of intellectual property rights with respect to claims on patents, copyrights, etc.

THE RESOURCES

Since 1980 there has been a growing library of books directly related to the history of computing, but the market has not been large compared to other texts, and it is rare that a book has seen a second printing or a second edition. Publishers have been reluctant to expend the necessary support for the up-front cost of printing and binding the initial print run. While several books have related to specific topics and resulted from conferences on the same topic, there have been few texts which provided a comprehensive overview of the topic. Michael William's The History of Computer Technology (1985) was extremely popular but the lack of a second printing has stifled the offering of university and college courses on the history of computing. Stan

[3] Galeazzo Ciano (1903-1944), an Italian Fascist leader, according to the electronic edition of "The Concise Columbia Dictionary of Quotations".

Augarten's Bit by Bit (1984) seems to have suffered the same fate.

The one publication which has stood the test of time has been the *Annals of the History of Computing*. Originally sponsored by AFIPS Press, the *Annals* has been actually printed and distributed by four different publishers over the past fifteen years, and is hopefully now well established within the IEEE Computer Society. The fifteen years of quarterly publications have provided the subscribers with insights into a wide range of subjects from early calculating devices such as Napier's promptuary to the sociology of state-of-the-art supercomputers. *Annals* has suffered from a strong orientation towards the English speaking world, though there has been a greater attention to the history of computing elsewhere in the past seven years. The history of computing in France was included in five separate issues, and one special issue has been devoted to the Canada. Individual papers have examined certain aspects of national computing activities, such as a recent extensive paper on the USSR. Also the *Annals* has become the primary journal for the publication of papers relating to Charles Babbage, his engines, and his other innovations. New textbooks are on the way, but what is really needed is a paperback that can be a supplement to the regular class textbook and that will be priced at such a level that it is not out of bounds for students to purchase as a second text.

HOW WILL WE USE HISTORICAL INFORMATION?

An examination of many of the central concepts of science from birth to maturity shows a perceived complexity versus time curve (fig. 1) which is roughly parabolic. In the beginning, the initial idea is often expressed in such complex terms that it is overlooked by the majority of readers. The one or two who can see through the veil of complexity simplify the concept, and express it in understandable terms, that in computer science can lead to implementation and application. As the resulting system is used and explored, it is again modified and enhanced to achieve a new zenith of complexity in that the level of applicability is high while the degree of understanding (seen as the complement of perceived complexity) has dropped once again to a low setting.

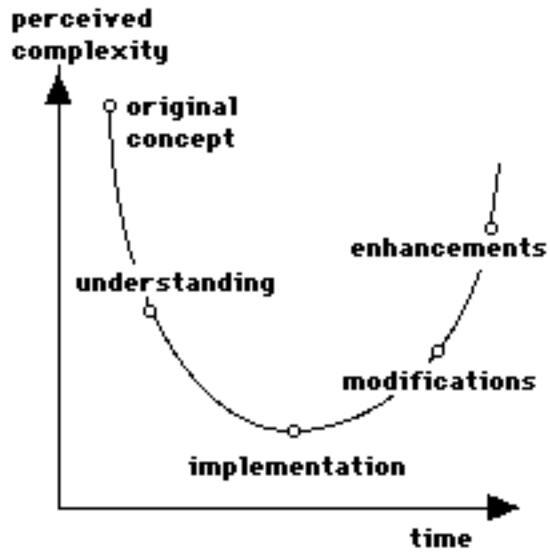

Figure 1. Perceived Complexity over Time

We have many examples of this phenomenon in computer science and I am sure that similar examples can be found in other sciences. The phenomenon of increasing perceived complexity in the latter stages of development creates a barrier against the overall study and presentation of an apparently complex concept. To better understand the concept we should return to the nadir of the perceived complexity curve to work outwards to comprehend the source of the concept and its evolution. To identify this point, we need to able to fall back on the history of the field and to be able to identify that point of minimal perceived complexity. Sometimes the solution is as easy (for a historian) as collecting the literature surrounding the concept and locating that first paper that "makes sense". Not necessarily the first *Scientific American* paper, but perhaps the one with the highest citation score. Such papers are the ones that should be used in class to introduce a subject.

HISTORY AS A SET OF CASE STUDIES

John von Neumann has been blamed by John Backus for the "bottleneck" in computing and the restrictions of the so-called von Neumann machine. Backus' Turing lecture in 1978, in which he introduced his concepts of functional programming (FP), was entitled "Can Programming be Liberated From the von Neumann Style?". In fact, Backus' understanding of the contributions of von Neumann was in error. Von Neumann recognized the advantages of parallel computation, but he also recognized that in his time the technology of architectural design was not sufficient to support truly parallel systems, that programming had not developed the techniques of multi-programming. Thus to solve the problems of the day he accepted serial or sequential computation -- the very technology for which he is now blamed! To complicate our understanding of the development of the computer, von Neumann allowed a myth to build around the origination of the concept of the stored program. The so-called EDVAC report of the work of several scientists was never completed and the first draft (von Neumann, 1947), which only ascribed authorship to von Neumann, was taken to be the original source of the idea. Over the years, von Neumann never took the time to dissuade his creditors of the actual genesis of the stored program concept. Similarly, Grace Murray Hopper was credited by some writers of her many obituaries with the "development of the programming language COBOL". History clearly shows that she would better be credited as the *grandmother* as the originator of FLOWMATIC, or as the *midwife* as a member of the organizing committee that recommended the evolution of a business oriented programming language, or as the *mentor* in her capacity as one who promulgated the language and possibly instigated the writing of the first two compilers. Over the years, Dr. Hopper never took any steps to dissuade believers of this misplaced credit, but neither did she personally claim the credit for the language. John Mauchly claimed credit for the invention of the computer, and for its construction in conjunction with J. Presper Eckert. Yet in 1973, the patents were invalidated by the Minneapolis District Court and "one, John Vincent Atanasoff" was, by law, given the credit for the invention of the computer -- at least in the US. Clark Mollenhoff (1988) went so far as to accuse Mauchly of perjury on the witness stand.

One of the concerns of today's education is the integration of the ethical concerns into our curriculum. From our three examples above, and many others that we can find in our history, we have a set of historical cases from our own profession for study by students which have a direct bearing on ethical studies in computer science. Here is a prime example of the application of episodes from our history to modern teaching in a field that is continually fleeing from its past.

HISTORY AS A TEACHING TOOL

Is history a *way* of teaching computer science? Is history a *part* of teaching computer science? Is there an advantage to going that *far* back? In fact all teaching is the teaching of history -- or at least the retelling of events and discoveries of both the recent and the far past, but not in the context of history. Learning is the understanding of discoveries and the application of self-discovery techniques to personally unexplored fields. Research starts with the study of the history of a topic and attempts to build new lines of discovery or development. Thus the questions above should be better stated in terms of placing the teaching of computer science in the environment of historical scholarship. The answers are most likely in the negative, but the preparation for teaching and the management of learning can take advantage of knowledge of the perceived complexity curve to select the appropriate point for creating a beginning point for understanding.

Is there a minimal body of knowledge of which we would expect computer science to be aware? Is there a litany of events, of persons, of machines, of software that should be engraved on the mind of each student? What test would we apply to determine whether a student has had enough exposure? Is there a training program that we can develop to assist faculty to prepare themselves to include historical facts in their course presentations and in the student's learning module? One part of the answer is of course that the expectations will vary with the home country of the student. A basic knowledge of the contributions of one's countrymen; the next stage should include the contributions of other pioneers that have affected the systems that they will use and study. It is just more than learning trivia -- the answers to the yearly questions in the ACM/Computer Museum Computer Bowl! More than just being able to make a reasonable score on the self-evaluation test created by Eric Weiss (1983).

Including historical perspectives in commonly taught computer science courses can bring some realism to the topic and at the same time provide some relief from the stultifying effects of purely abstruse presentations. For science students there is an additional advantage to studying history -- the opportunity to develop other skills such as writing and presentations. Rather than using history merely as an old story to be told, the history of computing has elements which can be used effectively to supplement and support the pedagogical aspects of the field. Starting too far along the development curve deprives the student of the understanding that concepts do not emerge full blown. Starting early on the development curve, and at the nadir of the perceived complexity curve, strips away the numbing, hubristic impact of enhancements and extensions.

In the obituary for Jim Wilkinson (Anon. 1987), pioneer numerical analyst, which was constructed from the e-mail comments of his colleagues, students, and admirers, there was a remembrance that has significance here:

> *I always looked forward to his lectures. The image I remember is of his lecturing in short sleeves -- sleeves held up by garters to achieve the correct length. ... <u>He had a great knack for working in an interesting historical anecdote that illuminated what he was trying to say.</u>* (Emphasis added)

Historical facts do have a place in the classroom, and in our ability to get across ideas to students in a meaningful manner.

CONCLUSIONS

Like a canoeist, we sit bobbing on the edge of an eddy, on the one hand, ready to be swept downstream to continue the fight against the flow of nature, and on the other hand, straining to achieve the chance to collect our thoughts in the still waters of understanding. This oscillation between pure and applied science swirls us around, the most effective progression downstream being accomplished by conserving our energy and analyzing the last descent, before we tackle the next obstacle. The *technolust* in our profession today discards the immediate past in favor of the latest toy; we do not pause long enough either to enjoy the tranquillity or to understand what we have accomplished. A well mounted hype that builds desire for glossy new artifacts has begun to win over reason and planning. As the amount of data left behind decreases, our ability to analyze the ongoing computing experiment and the problem grows more and more difficult. What *really* did work and what not? Is speed taking the place of ingenuity? Is faster so much better?

In the development of a set of requirements for the accreditation of computer science programs by the joint ACM/IEEE Computer Society committee in 1985, a period when the newspapers were having a field day with the repeated pranks of computer hackers, it was decided that every computer science curriculum ought to include the study of the ethics of computing. This was not prescribed to be included in a separate course, but could be integrated in courses across the curriculum. So it should be with the history of computing. History by itself can be very boring; but history in the context of a special subject brings that topic to life. This author has found that a student's appreciation of either compilers or early programming languages can be greatly enhanced (brought to life) by the showing of a video tape of interviews of the Fortran (Leeson, 1984) team reminiscing about their days together building the programming language Fortran and its first compiler. More and more live recordings of our pioneers are being made available; they can have a profound effect when used in courses other than history courses.

There *is* a place for historians in computer science and the study of historical elements of the field by students, at all levels, is legitimate. The historian cries out to the fleeing back of a computer scientist "pause to smell the flowers"! Let us work together to create a curriculum to further enhance the learning in our computer science courses through history.

REFERENCES

Anon. 1987. "James Wilkinson (1919-1986)", *Ann. Hist. Comp.*, Vol. 9, No. 2, pp. 205-210.

Augarten, Stan. 1984. Bit by Bit: An Illustrated History of Computers, Ticknor & Fields, New York, 324 pp.

Backus, John W. 1978. "Can Programming be Liberated From the von Neumann Style? A Functional Style and Its Algebra of Programs", *Comm. ACM.*, Vol. 21, pp. 613-641.

ComputerWorld. 3 November 1986. "The Computer Age", *ComputerWorld*, C.W. Communications, Inc., Framingham MA, pp. 14-18.

Hartley, Leslie Poles. 1954. The go-between, Knopf, New York, 311 pp.

Hendry, John. 1989. Innovating for Failure, MIT Press, Cambridge MA, 240 pp.

Leveson, Nancy G., and Clark S. Turner. July 1993. "An Investigation of Therac-25 Accidents", *IEEE Computer,* pp. 18-41.

Lee, J. A. N. 1994 "Why We Study the History of Computing", Proceedings, IFIP World Computer Congress, Hamburg, Germany, IFIP Transactions, A-52, Vol. II, North Holland, Amsterdam.

Leeson, Daniel. January 1984. "IBM FORTRAN Exhibit and Film", *Ann. Hist. Comp.,* Vol. 6, No. 1, pp. 41-48.

Mahoney, Michael. 1993. "What Makes History?", Second History of Programming Languages Conference, Cambridge MA, ACM SIGPLAN Notices, March 1993.

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

Mollenhoff, Clark R. 1988. Atanasoff: The Forgotten Father of the Computer, Iowa State Univ. Press, Ames IA, 274 pp.

Runyan, Linda. 15 March 1991. "40 Years on the Frontier", *Datamation,* Cahner-Ziff Publ. Assoc., Newton MA, pp. 34-57.

von Neumann, John. 1947. "The First Draft of a Report on EDVAC", reprinted in *Ann. Hist. Comp.*, Vol. 15, No. 4, 1993.

Wexelblat, R. L. (ed.). 1981. History of Programming Languages, Academic Press, New York, 758 pp.

Williams, Michael R. 1985. A history of computing technology, Prentice-Hall, Englewood Cliffs, N.J., 431 pp.

Weiss, Eric A. July 1983. "Self Assessment procedure XI: A self assessment procedure dealing with one part of early computing history", *CACM*, Vol. 26, No. 7, pp. 479-482.