USING COMPUTING HISTORY TO ENHANCE TEACHING

John Impagliazzo and John A.N. Lee
Hofstra University, USA; Virginia Tech and Radford University, USA
E-mail: cscjzi@Hofstra.edu; janlee17@verizon.net

Abstract: This paper focuses on a "history approach" as a way of teaching courses in computing. This project, funded in part by the National Science Foundation of the United States, is to use history as an ongoing theme throughout a course through the adaptation of relevant historical materials. The paper shows how the authors used historical materials to enhance course delivery. The expectation is that using history can make for a positive change in a course from one of sterile factual content to one containing dynamic interludes involving people, places, and events. In this manner, computing courses should be much more interesting. This mode of teaching should also stimulate more students (non-specialists) to consider computing as their major field of study.

Key words: History; Education; Case studies; Pedagogical tools

1. INTRODUCTION

Using history as a pedagogical tool should contribute to students' life-long learning experiences, encourage students from diverse and minority communities to appreciate computing, and increase student retention. Students should also gain a better sense of the nature of inquiry, the processes of innovation, and the human dimension.

In the beginning one of the major drawbacks to developing courses in the history of computing in educational institutions was the lack of reliable resources, trained teachers, and institutional support. While we believe that the case for using history in the normal compliment of courses, institutional support has been sadly lacking. History is not commonly a subject that leads to industrial development or has significant advantages in maintaining a

business presence in the marketplace. History is not a subject that attracts significant research grants (as compared to other grants in the IT field) and is not of great advantage to a faculty member when being considered for promotion or tenure. To a lesser extent computing education suffers from similar drawbacks as a topic of concentration in research universities.

Within a computing curriculum, the inclusion of a study of historical elements has the advantages that it will require computing students to "think outside the box" for a while, and particularly, outside the box defined by the computer perched before them. Assuming that a study module is more than a lecture style presentation and that students will "research" the topic themselves, history will force computing 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 has the advantage of requiring students to do some library research beyond the world wide web, to write more than comments in program headers, and perform some critical thinking that is not necessarily algorithmic—all subjects that are not the favorites of computing majors, but which need reinforcement. A subject such as computer science is not famous for its case studies as an educational methodology, but history may be able to provide the resources to foster this approach to computing education.

Why use history? While the tenet of this paper is the ability to enhance greatly the teaching of computing by using history as a medium, many other benefits exist that we cannot ignore. Among these are the concerns that unless we understand the history of the field, then we may be doomed to repeat the past. In many instances, the study of history can resurrect ideas and concepts that either progress overcame or were ahead of their time. For example, students who have grown up in the UNIX environment may be familiar with `lex` and `yacc` as means of language analysis, but when faced with another environment, they are unable to apply the more fundamental concepts of finite state machines to a similar problem. Though the center of computing is the inanimate objects of hardware and software, computing was the invention of pioneers whom we should neither forget nor deny their due credit.

History can also promote curiosity that overcomes the beliefs of many that "history is past" and has no modern value, or that science progresses to overcome its past. Unfortunately, in the eyes of professional historians computer history is too recent to be history and too recent to be forgotten. Chronological history is easy to present and readily available in many places, but like the history we were taught in high school it is without meaning unless it is related to other fields of study. History is mostly the

history of successes since the "winners" write most of history. Yet we could learn much from our failures.

Using history as an educational tool is giving students the ability to know how to build on the past and to follow Isaac Newton's testimonial that "If I have seen a little further than others, it is because I have stood on the shoulders of giants."

2. HISTORY IN THE COMPUTING CURRICULUM

It is appropriate that one make a clear distinction 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. A 1998 IFIP report produced by a joint committee selected from TC-3 and WG9.7 provided the resources for the construction of several different styles of history course in computing (Impagliazzo et al, 1998).

The report justifies the establishment of a history component in the computing curriculum by providing a framework for a curriculum and its supporting resources. It was international in scope and is not confined it specific computing disciplines. It also sought to raise the awareness of history and the manner in which people could use it to improve the study and practice of the computing profession. The report recommended guidelines on how history could be included as a part of computer education at the college or university level. The report suggested three ways of carrying this out:

- An integrated approach where history becomes a part of existing courses;
- A single course for computing specialists who have completed at least one year of study in their specialty; or
- A single course with no prerequisite open to all students.

These modes varied in the depth of presentation and the courses offered in the respective programs. The approach taken by this report was to propose a knowledge base of suggested historical topics, including current events. Irrespective of the mode of implementation, the topics contained in the knowledge base suggest a possible course framework. Teachers can customize the knowledge base according to the needs of their individual programs. For the integrated approach, teachers can incorporate the historical topics into existing courses. It is this latter approach that we consider further herein.

3. HISTORY IN THE FIRST COURSE

In any curriculum, the introductory course is the key to maintaining 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" [Tucker et al, 1991] these are generally designated as courses CS0 and CS1. While some may assume that the content of the literacy course is part of the high school preparatory repertoire in many curricula, such content is an essential element of a 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, and more to computing than what one can do on a modern computer.

The approach taken by these authors 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 particular 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 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. A tour-bus in a major city shows tourists where they might find primary sites; similarly, introductory courses show the locations and facades of the key topics within computing areas.

Abstract machines are easy to ascribe to their originators and this is one more opportunity to teach something of our history. For example, the study of compilers as a tool of computer scientists includes the study of finite state machines for lexical analysis. The study will also show that simple syntactic analyzers relate to 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 an example of how we have assembled the foundations of computer science from a variety of sources [McCulloch and Pitts, 1943] and conversely the applicability of computer technology to other fields. The history of programming languages starting from the early determination of 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 [Wexelblat, 1981, and Bergin and Gibson, 1996].

Following up on the claim that Alan Turing 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 [Turing, 1937]. While there possibly exist some emulators of the early machines, 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 [Hodges, 1983] and his many other contributions to the field of computing and to other fields such as biology. Not the least of his work was the code breaking at Bletchley Park during World War II. This is one case also where there is available a one-hour NOVA video program entitled *The Strange Life and Times of Alan Turing,* which can provide details beyond the capabilities of most teachers. A course on the fundamentals of computers and computing is by its very nature a course on the history of computing.

4. HISTORY AS A SOURCE FOR CASE STUDIES

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 to find details in the ensuing courses. This presumes that besides the teachers of the introductory course, faculty members in other courses are familiar with the history of the field. This raises the question regarding the requisiteness of teachers of technology being conversant with the history of at least the technology they teach. For several decades, computer science has ignored its history in favor of a forwarding-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, there is still need to look back at the histories of (say) operating systems, human computer interaction, or specific subjects such as object oriented programming. Moreover, most specialized textbooks do not contain a section on the history of the subject, and unfortunately,

where it is included it 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, providing a set of slides/overhead foils, or a PowerPoint presentation, 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 teachers can extract learning points to create a better course plan using history to teach the topic [Lee, 2000].

We can see the "proof" of the concept by considering the perceived complexity of a number of developments in our field. For example, the development of operating systems for personal computers from CP/M through MS Windows is an excellent case study that shows how attitudes changed over the years. CP/M, though extremely simple in concept was viewed as the domain of the microprocessor priesthood and thus extremely complex to the general user. DOS for the IBM-PC improved this perception, but the original operating system for the Apple Macintosh marks the point where the general user was accepting of the notion of a helpful operating system. Since that time, enhancements and "improvements" have generated a Windows operating system that some people regard as "complex" even though many use it. The "understandability" of operating systems for PCs improved from the time of the CP/M system to the Mac OS, but decreased thereafter. Similar trends appear in programming languages from FORTRAN to Ada, with a second overlapping curve showing the perceived complexity changes from the original C language to Java. Such secondary curves clearly indicate that someone decided to step outside the box and to solve the problem of perceived complexity.

5. HISTORY IN A PROFESSIONALISM COURSE

History can also be a source of case studies for use in courses related to ethics and social impact. While a 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 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 people have avoided the Therac 25 problems?
- Should computer pioneers have been more protective of their inventions?

The collection of cases is another of those under-developed aspect 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! However, we cannot expect teachers to go looking for it; that is the task of historians. We truly need to share our resources so that others can benefit from these positive experiences.

6. HISTORY IN THE COMPUTER ARCHITECTURE COURSE

The computer architecture course, a common part of most curricula in computer science, is typical of those in which attention to history is most important and revealing. It would be inappropriate to introduce students to computers without discussing the "invention" of the computer and discussing the question of who had the inspiration. Even a discussion of what constitutes a computer and how a computer is different from a calculator is essential. Here is an opportunity also for providing students with an overview of the meaning of the word computer, and to understand that in the days before the word computer was ascribed to a physical device it was first ascribed to people who ran calculators for scientists and engineers. While firsts are always difficult to identify, an opportunity exists within this course to give proper credit to national pioneers who were the first in the country of the students to either build or install a computer. Hardware is often the focus of computing history since it is easy to recognize, but contributions to theory and software also significant.

How many typical computing students have ever held, touched, or even seen a computer or its components? Since computer science or any of its related fields center on the software and applications aspects, few students hardly ever receive exposure to the electrical and electronic elements of a computer. Few have ever held resistors or capacitors, transistors, vacuum tubes, memory chips, floppy disks, or motherboards. Looking at the

generations of computer development is an opportunity to examine these artifacts and to begin to understand the space and heat problems of early machines. Disassembling components also helps in understanding.

Today's machines have a "customary" memory hierarchy devised in the early days. While it seems natural today, the history of the development of hierarchical memories as proposed in 1962 by Tom Kilburn and demonstrated with the Atlas at Manchester University brings out the priority of invention in the years before IBM System/360. Similarly, the development of the cache, proposed by Maurice Wilkes in 1965 in short paper, and first implemented at Cambridge University preceded the first commercial machine to use cache – the IBM 360/85. From these ideas came virtual memory, an idea that is inherent in most modern day machines.

The story of pipeline methodologies and dynamic scheduling, introduced in the CDC 6600 in 1964, "the First supercomputer" leads to explanations of multiple function units, score-boarding, time-shared pipeline, and the Tomasulo algorithm. From there it is a short step to RISC machines and the contributions of John Cocke, in the mid-1980s, that led to IBM Power 1 and Power machine series.

Behind the potentially sterile views of hardware and computer architecture are stories. We can use these stories to help students understand the progress of development and to understand better how things have come together as a conglomeration of elements. Ultimately, we can show how these events have coalesced into the machines of today.

7. STORIES AND INTRODUCTORY COURSES

So, how do we involve history in our courses? How can historical story telling make computing courses more interesting? Our focus here relates to introductory courses. These courses often take an "overview" approach and students attending them come from a variety of specialties. Introductory courses often have a heterogeneous mix of students requiring the instructor to take inventive measures to retain student attention. Using computing history is one way to do this.

One program that attempts to change the way students learn in computing courses is a project at Hofstra University where history is a theme used within an existing introductory course. The course, called "Overview of Computer Science", educates over two thousand students each year. Students learn the breadth of computer science and receive some exposure to a simple programming language such as JavaScript. The project, called Computing History Resource Adaptation (CHRAD), receives partial funding through the National Science Foundation of the United States. In selected

sections of this course, several teachers use history as a theme of presentation. For these course sections, students view two to three videos on the evolution of early computers, the history of software, and one on women and computing. Students also receive written summaries of these videos for reference. Students also explore computing history through various collections at virtual museums such as the Computer History Museum in California, the Charles Babbage Institute in Minnesota, the IEEE History Center, the American Computer Museum in Montana, and other museums worldwide.

The project has also developed other historical tools to facilitate a student's ability to research computing history. These include historical timelines and chronologies, and historical "slices" according to the knowledge areas found in the Computing Curricula 2001. In addition, the project developed other focused areas related to computing history such as the antitrust cases with Microsoft, the evolution of the transistor, and DNA and quantum computing. The project describes the latter two areas in an elementary manner rather than in a rigorous way to accommodate the abilities of the intended students.

The result of the CHRAD project is very positive. When compared to sections that do not use the history approach, students were able to identify computing pioneers and had a "human" insight into the computing world. Attendance levels for the history sections were almost perfect. All students, save a very few, completed the course and almost no student had failed. Students appeared engaged and interested in computing and because of the history theme, could relate computing to people and places with a temporal perspective. Computing did not appear to them as a dry subject of concepts and abstractions. Rather, computing became alive where the dynamics of life interwove with the subject content. Students seemed to connect and with the subject material and the course developed into a true liberal arts subject. For most students in this introductory course, the history approach has mitigated the "fear" of taking a computer course and has made them "connect" with the subject in a human way.

From a personal viewpoint, teaching computing from a historical perspective was and is simply fun. Bridging story telling with technical content forms a connection between the subject and the student that seems to form a true basis for lifelong learning. AH teachers should try this approach with their computing classes!

8. SUMMARY

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 the background history, authored in cooperation with a historian. Similarly, we need to encourage online virtual museums to go beyond being collections of images to providing "walk-through" galleries of 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,* that already contains (now in a digitized form through the IEEE Computer Society digital library) 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. Reluctance exists to fund extensive projects in the history of computing on 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 so computer historians are primarily "self-grown". Computer scientists with advice from historians of technology must lead the impetus for recording and recovering the history of computing to create the encyclopedia of the history of computing to support computer education.

REFERENCES

ACM / IEEE Computer Society, "Computing Curriculum 2001", (Final Report http://www.computer.org/education/cc2001/final/index.htm),

Bergin, Thomas J. and Richard G. Gibson, 1996, *The History of Programming Languages - II,* Addison-Wesley Pub. Co., 864 pp.

Godfrey, M.D., and D.F. Hendry, 1993, "The Computer as von Neumann Planned It", *IEEE Annals of the History of Computing,* Vol. 15, No. 1, pp. 11-21.

Hodges, Andrew, 1983, *Alan Turing: The Enigma,* Simon and Shuster, New York.

Impagliazzo John, et al., 1998, History in the Computing Curriculum. IFIP TC 3 / TC 9 Joint Task Group, http://www.hofstra.edu/pdf/CompHist_9810rprt.PDF (See also: *IEEE Annals of the History of Computing,* Vol 21, No 1, pp. 1-15, January 1999.)

Lee, J.A.N., 2000, "Perceived Complexity: A Conjecture Regarding the Life Cycle of Computer-Related Products", IFIP World Computer Congress 2000, Beijing China, August 2000.

McCulloch, Warren S., and Walter Pitts, 1943, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bull. Math. Biophysics,* Vol. 5, pp. 115-133.

Tucker, Allen B., et al., 1991, Computing Curricula '91. Association for Computing Machinery and the Computer Society of the Institute of Electrical and Electronics Engineers.

Turing, Alan M., 1937, "On Computable Numbers with an Application to the Entscheidungs-problem", *Proc. London Math. Soc.,* Vol. 42, pp. 230-65.[6]

Wexelblat, Richard L., 1981, *The History of Programming Languages*, Academic Press, 758 PP.

Williams, Michael R., 1997, *A History of Computing Technology,* Second Edition, IEEE Computer Society Press, March 1997, 440 pp.

Zemanek, Heinz, 1979, "Al-Khorezmi: His Background, His Personality, His Work, and His Influence", *Symp. Algorithms in Modern Mathematics and Computer Science,* Urgench, Uzbek, USSR, September 1979.

ENDNOTE

The National Science Foundation of the United States supported part of this work.

[6] This paper defined the concept of the Universal Machine.