APPLICATIONS OF TECHNOLOGY IN TEACHING CHEMISTRY
An On-Line Computer Conference
June 14 TO August 20, 1993

PAPER 9

Staff Development is the Biggest Cost in Computing: Ask For Released Time!

David W. Brooks, University of Nebraska-Lincoln, Lincoln, Nebraska 68588-0355   E-mail: dbrooks@unlinfo.unl.edu

SCHEDULE: Short questions on this paper: July 8, 1993
           Discussion of this paper: July 20 and 21, 1993

ABSTRACT: Conversations about computer use often include comments about access to hardware and software. In reality, the cost of a powerful computer with content specific software is now a small fraction of a teacher's annual salary. In spite of this, instruction regarding the use of computers -- particularly as thinking agents -- lags. The primary reason for this probably is that the total amount of time needed to become importantly conversant with computers and their software is very significant such that the relative salary cost in this area is still on the order of one or two salary years. Persons participating in an electronic conference probably have paid or begun to pay much or most of this

---

One should always be suspicious of papers presented at meetings that make excessive use of first person pronouns.

Because my kinetic data required analysis, I learned to program on an IBM mainframe using Fortran when I was a graduate student during the early 1960s. When I taught thousands of students in general chemistry, I maintained a very complex grading scheme using Fortran and then APL. My first PC was an IBM, purchased on an NSF CAUSE grant. In 1987, it became clear that a Macintosh computer with HyperCard was going to be the best and easiest way to control videodisc players; I switched all of my personal computing from IBMs and Amigas to Macs.

One evening in 1989, after a half-bottle of wine and a challenge from a colleague at a meeting far removed from home, I went to my hotel room and created a draft nomenclature program. It gave a name and you responded with a formula; it gave a formula, and you gave it a name. It was a drill and practice affair that included aural output of the name. It took about 2 hours to get a working version ready. I demonstrated it during my talk the next day. My skills with HyperCard had developed to the point where a presentable hack was possible after just a couple of hours of effort.

Just over two years ago, while away at a meeting, a colleague led my seminar group and talked about a CAI program to teach oxidation numbers. Upon my return, each of my graduate students and associates suggested that I take a stab a creating a similar or better program. Their estimates were that it would take me from 4 days to 3 weeks, not several person years. Because I was able to start from the nomenclature "hack," it took only one day. I was feeling very smug about my success.

My strategy in creating chemistry teaching materials had become a formula process. The student would type in a name or a formula or some other part of an exercise; my program would crunch and determine answers for that input; the student would input her answer; my program would compare its answer to the student's answer; and feedback would be provided, either a 'good job,' or an explanation of the steps used by the program to get to its answers. About two months after creating the oxidation number program, I had the rudest awakening of my professional life. A clever person would trash the back part of my program and just use the front part of it to get an answer!

However upset that notion may make you -- using software just to get an answer -- that's how the world works. Teachers, especially college science teachers, forget about world pragmatism. Also, up until now getting an answer has not been simple. For the most part, you would need to find an expert to answer end-of-chapter chemistry questions if you didn't have personal expertise.

Since that shattering time I have worked on completely different programs. Instead of teaching someone a topic, I try to get the program to perform as much of the topic skill as possible. My current stoichiometry program, for example, can earn someone a B grade in nearly any general chemistry course.

Up until 1991, I was doing what everybody else was doing: trying to create materials or use strategies that helped people acquire the 'old' skills either faster or better. Anything that might work was fair game, from a new teaching strategy or lab experiment to a computer program or simulation. It is now absolutely clear to me that professionals use and will come to use computer tools in ways unheard of just a decade ago. The nature of professional activity in essentially all professions is undergoing fundamental change.

A generic way of describing the most important technical skill that a professional can have is to be able to use a computer as an intellectual partner. It follows that computer partnering is the most important professional and technical skill that a teacher can have. Every day that a student goes to school at any grade level and does not see his or her teacher use a computer as a tool is a day that the student has gotten a year 1970, not a year 2000 perspective of intellectual activity.

If you have my job -- professor of chemistry education -- and you really believe as I do that a massive revision in intellectual processes is underway, then the nature of the task at hand changes enormously. For the last three years, I have been encouraging computer tool use in a broad variety of ways. I have created many software tools, and embraced many others.

As I look around, I see very few teachers doing this? Why?

After some time thinking about this problem, one dimension of the answer is becoming quite clear. Most chemistry teachers have neither an idea of what software is available nor what using that software means. We are not spending enough time in addressing this deficit. Said another way, we are not spending nearly enough money in this area. We have full time jobs. We're busy. The task requires an enormous amount of time.

When companies study successful implementations of computers, they discover cost patterns. For every hardware dollar, there is another software dollar and perhaps 20 cents in repair, but also three to five dollars in training! Last semester I modified a graduate course in such a way as to make numerous site visits to schools teaching computer use in my region. The computer technology personnel ALL indicated that the most serious shortage in resources was neither hardware nor software but time to train faculty. Mind you, each has large hardware and software needs.

Upon reflection, I was neither an early convert to my current thinking, nor was I easily convinced about the impact of computers. My first computer experience with PCs was grudging -- my colleague, Tom Tipton, said something like "you really ought to look at VisiCalc for TA scheduling," and I looked back at him and muttered something about having no time. After all, I was among the legions busily engaged in teaching the curriculum of the 1960s as modified by the hand held calculator of the 1970s to the student of the 1980s. But I started messing around a little bit with a PC, and then more, and then more. Since I left the chemistry department, my teaching assignment has included teaching graduate courses in applications of computers -- so the time spent in this area has quintupled. When I try to add up my hours spent, the number is enormous. I have probably spent four or five person years of my life at a computer terminal (8,000 to 10,000 hours -- that's where I sit at this moment!)

Although my conversion to a computer-tool-use view of intellectual activity is essentially complete, the time that I have spent thinking about the nature and significance of the problem as it impacts upon instruction has been relatively brief. This paper at this conference represents the first time that I have committed some of these notions to paper. I've taught HyperCard scripting to over 150 graduate students (a very large number for such a specialized course in my college, especially one with a reputation for requiring considerable work), and given Chautauqua workshops to a still larger number of college faculty. Most of these experiences do not seem to seem to have lasting impacts. People come and engage for a while, but the total time required to 'get good' at HyperTalk scripting and designing is much larger than the time available during workshops or even during courses. Although a few of the students have gone on to create 'commercial' software, those succeeding at that level largely came to the course with a great deal of prior knowledge. Why are the impacts small? To get to be good, lots of time is required. Lots of time! Very few have spent enough advanced time, or go on to spend enough time. That's not a surprise. It takes a great deal of time to learn how to use my word processing program and my spreadsheet, too. I can easily afford to spend a couple of profitable weeks just playing around with my spreadsheet.

Chemistry educators still see computers as gadgets not unlike calculators. Calculators increased the attrition rates in chemistry not because they made problems harder but because they made it possible to tackle harder problems! One had to know more chemistry to be effective in the world of the calculator. Today my software will compute excellent answers, sometimes outrageous answers. As more databases are tied in, fewer outrageous answers appear. You can no longer ask my programs how to make 50 M NaCl and get an answer without also getting questions about the reasonableness of your question. The currently available programs, the successors to those distributed on the ChemSource CD-ROMs, will check your chemistry at that level. As we start to include neural networks into our software, they'll be better still.

The computer education literature speaks to the difference between top-down and bottom-up innovations. It is generally thought that bottom-up innovations are more successful. This is interpreted in many ways, the most common grounds being in terms of greater morale, acceptance of change, etc. Better levels of teacher knowledge is rarely considered to be a reason. Of course, in top-down innovations the supply of time and resources allocated for faculty training is almost always much smaller than the need. Perhaps, in bottom-up innovations, this cost is borne by the participating faculty. Perhaps bottom up innovations have a better track record simply because there is more knowledge behind them.

The purpose of this paper is to emphasize the seriousness of the training problem. We have to begin with ourselves, not our students. The persons contributing to and participating in this conference are probably among the technologically most well prepared faculty. Are we well enough prepared? The answer is almost certainly, 'no.' Time for faculty development -- very expensive time -- rarely is allocated. After all, we're already supposed to be experts. Indeed, we are the ones to whom campus administrators usually turn to provide training for other faculty.

But the need for training at a time of great transition is enormous. There are two ways for us to bring this point home. First, every academician reading this paper could request time to become better prepared to use and teach the use of computer tools. Ask for released time. Our employers, Federal and state agencies, foundations, and others must help us find the time and money. Second, not only should we as teachers model computer use in our classrooms and offices every day, but we should come to expect the same modeling from every school teacher, from kindergarten on up. Yes, that includes art teachers, music teachers, and physical education teachers.

Suppose one has a tool that 'does' stoichiometry. What must one know about stoichiometry to be able to use that tool effectively? Will effective use of the tool change and evolve? How little training can one have and still be an effective tool user? When do you need an expert -- someone who "understands" the tool fully, perhaps even to the point of knowing how the innards of the tool function? We need to begin developing some creditable answers to these questions. There is one thing that is quite clear to me, however. Time spent in having students acquire skills that pit them against the software now available is wasted time. There are drill and practice program that really help to build skills in certain areas -- gas law problems, mole problems, assigning oxidation numbers, naming compounds and writing formulas, etc. The existence of these programs is prima facie evidence that the skill probably is not a worthy skill to teach!