MacLaboratory for Psychology: Successes, failures, economics, and outcomes over its decade of development

DOUGLAS L. CHUTE

Drexel University, Philadelphia, Pennsylvania

The history of the development of MacLaboratory for Psychology offers insights into both problems and solutions in academic software creation and implementation. Superficially, MacLaboratory for Psychology has been very successful, having garnered numerous acclamations, including two EDUCOM awards. It is used on thousands of Macintoshes at universities, hospitals, schools, and other institutions worldwide. However, there have also been some shortcomings to its use. In a candid discussion, successes, failures, the economics of software development, and student outcomes are addressed. Specific examples from the software as it has been used by over 10,000 undergraduates at Drexel University provide guidelines for future academic software in such domains as multimedia use, psychological and scientific education, classroom implementation, faculty research, and software support and maintenance.

Background

The first software iterations that eventually became MacLaboratory for Psychology were originally attempted on the Apple II in 1982 while I was at the University of Toronto, but they were rapidly abandoned because of my lack of programming ability, the lack of institutional support, and the limitations of primitive command line technology. MacLaboratory for Psychology formally came into existence in 1984 with the advent of the 128K Macintosh (Chute, 1986) and continues to evolve on succeeding architectures for Macintosh, A/UX, and PowerPC, the Apple/IBM RISC-based machine. It was based on a successful curriculum pioneered in the 1960s by the noted teacher of psychology, Robert S. Daniel (Daniel & Chute, 1972). The core tenet in his teaching was, and continues to be, to use a hands-on process approach to learning the essentials of experimental methodology. By enabling students to use the computer as a flexible research-grade laboratory "apparatus," we could have them design, conduct, and report on unique scientific investigations. Unlike in physics, chemistry, biology, or other traditional undergraduate labs, the students were not simply replicating age-old classic experiments whose results were, as often as not, fudged in order to "get the right answer." Instead, they were conceptualizing and developing research protocols, which in the best of circumstances were unique and original. This appears to be a better way of developing scientific thinking and practical experience in a realistic research environment (Chute, 1990a).

For their commentary and ideas, I thank Professors Tom Hewett, Walter Schneider, and Seth Roberts. MacLaboratory software has been supported by Apple Computer, The Pew Charitable Trusts, The Solon Foundation, the Hanspeter Albisser Stiftung, and Drexel University. Correspondence should be addressed to D. L. Chute, Professor of Neuropsychology, Drexel University, Philadelphia, PA 19104.

A primary requirement for the core software in such a curriculum is that it must truly function at a research-grade level. The application Reaction Time, for instance, collects millisecond accurate responses for a variety of input devices and displays complex "photographic quality" images in as little time as a single 16-msec screen refresh (Westall, Perkey, & Chute, 1986, 1989). Another requirement is that students should be able to use such an experiment editing tool with a simple cut, copy, paste, click, and drag interface so that they will not be distracted by the technology or required to have arcane programming skills. The flexibility of the resulting software has allowed it to be used in many courses throughout the undergraduate and graduate curriculum, and its comparative ease of use has led to its adoption or site licensing at thousands of institutions worldwide, not just for teaching psychology but also for research protocols in cognitive science, neuroscience, biology, medicine, and business.

The current version of MacLaboratory, 2.1, contains over 600 MB of applications, research protocols, clinical protocols, tutorials, teaching materials, lab manuals, movie clips, multimedia, and documentation. It contains fairly typical examples of most types of academic software, such as tools, simulations, study guides, animations, reference materials, and tutorials (Chute, 1990b; Chute & Daniel, 1990a, 1990b). The senior product managers for this version were Robert F. Westall and Scott Quillen, with additional programming done by Ryan Sandridge and Joe Barisa.

By many criteria, from positive student outcomes, through national and international awards, to wide distribution and acceptance, MacLaboratory for Psychology has been a success. At the risk of sounding maudlin, however, there are notable failures, shortcomings, economic limitations, and other less than satisfactory results that a look

at the decade or so of MacLaboratory's use can illuminate. In this paper, I will try to candidly discuss both failures and successes, with the hope that our experience may generate some useful hypotheses about the use and development of academic software. Please do not infer that there are any results to be reported in the scientific sense.

Economics

Over the past 8 years, MacLaboratory salaries for programmers, software testers, and other production personnel have grown to about \$100,000 per year. In order to supply quality software that successfully runs on an ever-increasing number of Macintosh platforms, systems, printers, peripherals, and so forth, an additional \$50,000 or so of the latest hardware and software per year are required. For example, if we had not acquired new hardware, we would not have known that one popular Macintosh had the left and right channels for sound reversed. Apple was not aware of this, because normally it does not present a problem—that is, not until you try to conduct a dichotic listening experiment! Service, maintenance, training, delivery, and product fulfillment require another \$50,000 or so a year. MacLaboratory expends no resources for advertising. For a basic product that sells for \$189.90 with discounts for larger site license users, that pricing typically results in an annual loss of a few thousand dollars, which my family and I have been able to consider an expensive, though rewarding, hobby. If we had been business people, presumably, we would have figured out how to charge the right price. Initially, pricing at “less than \$40” was suggested by Apple because they believed that would be low enough to encourage students to make the legal purchase. With the exception of a few graduate students setting up their dissertations, I don't think a student has ever bought a copy. Such low pricing was controlled for a time by agreements with distributors such as Kinko's, and it has increased haphazardly since. MacLaboratory's small losses are to some extent offset by equipment donations from Apple Computer, as well as research and development grant support—for example, from the Hanspeter Albisser Stiftung of Zürich.

These economics of psychology software development are not peculiar to the Macintosh. Walter Schneider has developed a conceptually similar psychology research environment (MEL) for DOS-based machines, and in a recent discussion, we concluded that the economics for both platforms were remarkably similar, with MEL supported by the Schneider Family Foundation for several thousand dollars a year (Schneider, 1992, personal communication). Schneider also made a very interesting observation that could serve as an important new direction to permit the discipline to capitalize on this past work, cease reinventing the “software wheel,” and provide for a more broadly based development and advancement of the “art.” He observed that the National Science Foundation supports about 100 psychology laboratories. Overall, federal funding for psychology research amounts to about \$68 mil-

lion. MEL and MacLaboratory support 2,000–3,000 laboratories for a total development cost of about \$2 million. In very round numbers, this is about 30 times the number of labs that the NSF supports, although we recognize that research apparatus and the personnel needed for building and running it are only a part of federally supported costs that could be addressed by better software. Perhaps our software has also been “democratizing,” in that records of users indicate that many smaller institutions with less infrastructure seem to have become empowered beneficiaries. The implementation of MacLaboratory for Psychology has been the basis of numerous NSF equipment grants to other universities; personally, I have never been successful at this, perhaps because I write poorly, or perhaps worse because the necessity of a commercial aspect to keep the development process alive has unfortunately tainted my professional acceptability. Our otherwise successful grant history might lead to the suspicion that our exceptional and repeated failures at securing various types of NSF support for software development represent an unwritten policy, or singular prejudice on the part of our peers.

The positive side of the current economic state of affairs for experimental psychology software is that the major packages are basically at the point of breaking even. The downside is that we do not expect much growth, for most potential adopters of new technologies have already made their commitments. Future growth is necessary to expand the capabilities of the software tools for research, but most growth is expected to occur by proliferation of student-based teaching laboratories. It is unclear whether students can somehow be expected to subsidize software sophistication for research. Consistent with the Macintosh culture, MacLaboratory has not electronically attempted to regulate copying, but rather depended on such devices as software registration and the ethics of users to limit losses. Unfortunately, this may not be as effective as we would hope, and the situation seems to have deteriorated in the past year or so as research and departmental budgets have been tightened at most universities and colleges (Chute, 1991). I would note to their credit that our major users have been scrupulously correct in monitoring and ethically licensing and using the software.

Software Development

In the earlier years, it was part of Drexel University's mission to make a major investment in the development of software for academic use (Perkey, 1986). Although significantly less involved now in courseware development, the university primarily underwrote the expenses of the first iterations of MacLaboratory software through grants from the PEW Foundation, Apple Computer, and its own resources. Drexel University pioneered the architect-builder model, which was based on the astute insights of Tom Hewett, Banu Onaral, and others (Hewett & Perkey, 1984; Onaral, 1990). In this model, the faculty member served as the architect, with a critical mass of professional programmers working in a software development group

serving as builders. This iterative design approach was very successful; it eventually produced a large number of programs in many disciplines. For example, at one time about 40% of all software packages distributed by Kinko's Academic Courseware Exchange and its successor, Intellimation, were authored at Drexel University. Drexel University, like most other university developers, has discovered that this strategy is too expensive and too narrowly based to be cost effective in itself. For example, the Kinko's and Intellimation distribution systems provide a return of about 10%. Thus, a package that sells for \$80.00 returns only \$8.00 in royalties. Even the most successful packages only return a few thousand dollars, which hardly buys the computer for the faculty member, let alone offsets the true development costs. Another problem has been the difficulty in retention of high-quality programming staff. Most of the professionals and a number of the student programmers who have ever worked on MacLaboratory software eventually ended up employed by Apple Computer.

Perhaps a more detrimental effect on continued university development was the needed staffing and expenses associated with ongoing maintenance of existing software. Without exception, this could not be managed in a timely, professional, or business-like way, and eventually the development efforts languished and all too quickly became dated. In some instances, the university and the faculty were embarrassed when dated products that continued to be sold crashed under new systems or architectures. Many are no longer in use, even on our own campus. Faculty who had successful packages could not supply the telephone or business support that many users expect and deserve for their investment, and many faculty felt they had risked enough professionally in even developing a package, let alone in devoting the time and effort necessary to ensure its continued survival. Maintenance and service requirements seem to have been the most overlooked components in the production of viable and valuable academic software, and ultimately they became the Achilles heel for the architect-builder model, which had created so many products. To offer a quantitative, though subjective, view of the problem of maintaining viability, I would say that only 6 out of nearly 200 programs are still viable—that is, that are used at all.

Software Distribution

Administratively, Drexel University could not supply the infrastructure or effective administration to handle the continuing needs of users of the academic software that it had produced. For example, orders for MacLaboratory arrived, yet they remained unfilled because no one in the university had the job of servicing such accounts. As in most institutions, the computer services group was in some respects detached from the direct academic mission, which resulted in various failures of communication. I suppose we thought that such things would look after themselves, much as reprint requests do, but the demands that disk copying, documentation, mailing, and service place on

individual faculty and departmental financial resources or personnel are not justifiable. To illustrate this problem—MacLaboratory spends about \$10,000 a year on telephone calls and postage for mailing about 10,000 floppy disks and documentation, an expense that I could not reasonably expect to be absorbed by my department. Drexel University also had difficulties in establishing an appropriate accounting and financial infrastructure, difficulties that are probably shared by other institutions. For example, checks for software payments that did arrive had no appropriate place to fit in the accounting system, and in the case of MacLaboratory, they languished uncashed in a desk drawer for upwards of 2 years. The university eventually recognized that it was not equipped to be in the academic software business, and as in the case of MacLaboratory, Inc., it permitted faculty developers to distribute their intellectual properties through other means. Consequently, Drexel has naturally migrated to supporting large campus-wide projects such as networking and has moved away from discipline-specific software development and distribution. It is not clear to me what we will do with a network in the absence of continuing discipline-specific development.

MacLaboratory had been distributed by Kinko's and would probably not have survived except that my wife, Margaret Bliss, decided to distribute the package through a separate corporation, MacLaboratory, Inc., effectively recovering the 90% that had gone to Kinko's and retaining it to fund the continued survival of the software. We hope our users have benefited from the improved maintenance and support that this approach afforded. Its main limitation, however, is capitalization for new product development, especially in such expensive areas as multimedia integration and production. To provide capitalization, we directed some effort at involving the traditional college textbook publishing industry in the exploration of technological information dissemination systems (Chute, 1987, 1989). Together with John Noon, publisher of notable academic software information products such as Syllabus, The Higher Education Product Companion, and Query, we founded a Software Authors Association as part of the Textbook Authors Association and provided workshops and seminars for authors as well as senior management in the publishing industry (Noon, 1989). Almost everyone seems to have appreciated our efforts, but the fundamental economics and intrinsic conservatism of the industry favor a wait-and-see approach. Alternative publishing channels are consequently emerging from the technological side, as, for example, companies such as Intellimation and Voyager.

Publishing

Bob Daniel's original curriculum had had an accompanying student's laboratory manual that was originally published by Wm. C. Brown, later by the University of Missouri Bookstore, and later again by Kendall/Hunt. For the first computer-based version of the course, a manual was desktop published on the 128K Macintosh and printed

by Kendall/Hunt for 1985 classes. Apparently they viewed the project as marginal and it was dropped. In 1986, Kinko's published it by xerography locally but couldn't manage to supply a growing user base on other campuses. In 1988, McGraw-Hill published the manual and instructor's notes and in 1990 produced a second edition. Unfortunately McGraw-Hill has not managed to use the materials or the software potential effectively, either in conjunction with a text or in any way other than as a potential giveaway to secure an adoption. No doubt they have made considerable profit on 20,000 or so sales of an overpriced manual, but they have not seen any need to provide sustenance for its continued development. Only about 10% of the software users devoted to classroom instruction actually use the laboratory manuals in teaching. The laboratory manuals are useful when a department first decides to install the software in its curriculum. There are many problems attendant upon implementation of computers on a large scale, and the lab manuals get things started with a reasonably complete package. More experienced users end up customizing and developing their own materials.

I had expected, naively, that some publisher would see the potential of MacLaboratory's large established user base as a sufficiently valuable, reasonably loyal, pedagogically committed, technologically capable, ready-made foundation on which to build appropriate aspects of Information Age technologies. Initially the publishing industry had to overcome its view of software developers and users as Luddites and book burners. Although some now profess a vision of the technological future, the fact remains that they correctly see that there is no profit to be made in academic software as things now stand. For example, publishers perceive increasing demands by academics for more giveaways, making it less likely that quality software products can be sold, especially when the usual model of the student-purchased textbook does not apply. To ensure the continued development of valuable and viable software and curriculum solutions, MacLaboratory is faced with having to become its own publisher. There is a technological opportunity that may advance psychology teaching, should this eventuality occur. A consortium of users could emerge and through an electronically based custom publishing model provide a broad based and multifaceted library making individual expertise and scholarship available to all.

It is my perception, however, that the academic psychologist is more independent and less collaborative than is the case in other disciplines such as physics, for example, where sharing of resources like supercomputers has become more commonplace. Perhaps a role for the Society for Computers in Psychology (SCiP) might be to promote such collaboration for both teaching and research endeavors. The alternative is a continuation of individually developed cottage industry software, but my sense is that psychology as a discipline needs to find ways to advance beyond this stage to professional and powerful productivity tools.

Acceptance and Implementation

One of the major criticisms of existing psychology software is that it often seems idiosyncratic either in content or to the institution where it was developed. Many faculty believe that they could do a better job, but typically, as the magnitude and realities of the task emerge, the project ends up on the shelf, or at best is only used locally. In a very informal review of Macintosh software or code published in *Behavior Research Methods, Instruments, & Computers*, I could find little evidence that many of the "products" older than a year or two were still in use. This is even true of some MacLaboratory-developed products like Controller (Chute, Gaeman, & Ziegler, 1987), which failed to be truly "research grade," lost us tens of thousands of dollars in development costs, and found scant alternative use. Another failing MacLaboratory product is ColorCard, a HyperCard extension with an elegant color and windows interface. In this case, the tool was generally useful enough so that competing commercial companies like Claris outstripped its development and capabilities.

The upcoming generation of psychology authoring software functions more like a tool kit than a canned tutorial, thus permitting broad and comparatively easy customization even by nonprogrammers. Object-oriented programming environments such as HyperCard should be encouraged, to facilitate customization. The primary difficulty with such environments is that by their nature they cannot supply the programming rigor to be truly research grade. Some programs, like Barbara Tversky's award-winning cognitive package in HyperCard, can, however, provide reasonable teaching examples. Even these suffer from the need for ongoing, and annoyingly never-ending, maintenance and support. For academics with busy schedules, there is a conflict between the power and flexibility of tool-kit software and its required investment in time and implementation, and ready-made software requiring less user effort. In either case, classroom implementation is much more difficult, time consuming, and expensive than the hardware manufacturers would generally lead one to believe.

Institutions in the past have borne development and implementation expenses for prestige, but this type of commitment is waning nationally, probably for reasons similar to those of the Drexel experience. With it is also waning the modest institutional commitment to honor faculty developers with merit, tenure, or promotion for software projects. It may be appropriate for psychology as a discipline to consider how academic computing can be viewed more successfully as a professional and scholarly activity.

The difficulties of implementation of new technologies in either teaching or research environments vary from institution to institution. I will illustrate some of the major considerations from MacLaboratory's experience at Drexel, as well as provide my perception of the experience of representative users such as Dave MacDonald, University of Missouri; Denise Park, University of Georgia; John Flynn,

Baylor University; Alester Krames and Jeff Graham, University of Toronto; and Ed O'Brien, Marywood College, who have been generous with their time and creative input.

In teaching, MacLaboratory for Psychology has typically been used as a laboratory component in introductory courses, or in the beginning courses in experimental psychology or research methods. Some modules have been adapted locally for courses in neuroscience, cognitive, abnormal, and sensation perception, mostly at the undergraduate level. On the graduate level, the research applications, particularly Motor Skills and Reaction Time, are generally used for thesis and dissertation projects. At Drexel University, about 1,300 students a year take the introductory course, which is split up into laboratory sections with 20 computers for classes that consist of 2 h of lectures and 1 h of laboratory work a week. An undergraduate multimedia laboratory for Abnormal Psychology using MacLaboratory's Psychological Disorders stacks is scheduled in the same facilities in a similar course format.

At Drexel, our first teaching lab was built and controlled directly by the psychology department; it consisted of 128K Macintoshes and printers supplied by a grant from Apple Computer. This hardware quickly became limiting, and I was able to secure another grant from Apple to upgrade these machines to the Macintosh Plus a couple of years later. Initially, I had felt that departmental control of our laboratory facilities was important; after all, that was the way things had always worked. More hardware upgrades to Macintosh IIci computers, system upgrades, and courseware upgrades inevitably became necessary and were simply beyond the scope of the department's ability to keep managing. So, with some trepidation, we sold our hardware and moved the classes into the general university clusters. Of course, problems of scheduling, control, and so forth do exist, but these are relatively minor annoyances in comparison with the relief from hardware and software service, management, and maintenance. We gained access to higher level machines as well as file servers and networks. We learned not to bother with printing except to disk. The general principles that seem to apply at Drexel and at other sites might be summarized thus: trade departmental control for university maintenance, and purchase fewer high-level computers instead of many low-level machines; corollaries are that hardware is the cheapest part of any implementation, and that, in the Macintosh world, one should not buy less than a 68030 machine with 5 MB of RAM, 256 levels of color, an 80-MB hard drive, and a CD-ROM drive.

Another implementation problem affecting both research and teaching users concerns training on computer basics for TAs, faculty, research assistants, and students. To some extent this problem has decreased over the years as people have generally become more familiar with personal computers. To be able to take full advantage of the software, however, does require some expertise, as well as an investment in time. Perhaps workshops coordinated by professional groups like the Society for Computers in Psychology are required, as opposed to the more com-

mercial training programs offered for such packages as MacLaboratory or MEL. Fortunately, with the Macintosh, sophisticated research or teaching protocols can be made with simple click-and-drag and other graphical user interface commands. But nearly any project takes more time than is anticipated. MacLaboratory was designed for flexibility without the need for any of the programming skills that are required on command-line-based machines, and noticeably absent for the average student or faculty member. Inevitably, some sites using our software wanted major extensions or customizations. MacLaboratory has not been entirely successful in responding to such needs. Users are dependent on our development cycles and we are dependent in turn on Apple's. Even though MacLaboratory, Inc. is an Apple partner and seed site for many beta testing programs, limited resources make it hard to keep pace with the dynamic nature of the enterprise. Another difficulty is the variability in machine environments of our users. About one third use black-and-white Macintoshes, and the rest use color. About half use HyperCard 1.2.5 and System 6.x and half use newer versions. The comparatively slow process of software development in general and rigorous quality control to handle different machines and systems make a centralized model of development impractical. As a result, some sites have arranged for their own personnel to develop teaching and research materials. They have basically used four different models to accomplish their goals: (1) They access a local software development group and make modifications, with permission, to our resources. (2) They hire as consultants MacLaboratory programming staff. (3) They use the basic interfaces and scripts of the more open-ended components like the HyperCard stacks or Reaction Time application, and develop consistent looking new functionality or content. Or (4) they create experiment files and/or spreadsheet macros that use the research applications as run-time engines. It would seem desirable to make such work available to the broader community to build an ever-expanding base from which others can build, but such mechanisms are not really in place. Perhaps software archives like macpsych at St. Olaf's College (Huff, 1992) can provide such services, or distribution arrangements can be made through MacLaboratory, Inc.

At Drexel, we have tried some implementations that were not particularly successful and perhaps can be summarized as involving an overuse of the technology that had the undesirable effect of decreasing the role of the teacher. In lectures, for example, it is not too difficult to use simple HyperCard stacks to present lecture notes and materials, and these can then be put on a file server for general student access. This works, but it can have undesirable effects. Perhaps students go to lectures less because they know they can always pick up the notes! When you add laser disk players, and other such accoutrements, your classroom system becomes complicated and fragile (i.e., it requires long set-up time and fiddling, and it may not work right); this may not be worth the value added, in comparison with the use of a few overheads or

the play button on a VCR. I find that I can become so busy with the technology that I engage less in informal classroom chat. QuickTime movies and other multimedia projects are expensive to develop, and without a CD-ROM or optical delivery system, they are hard to put directly into the hands of students. In experimenting with hypermedia, we can supply knowledge "exploratoriums," but we have found that it is unwise to remove the structure and pedagogy a teacher can provide. Students without such structure at best acquire "encyclopedic" collections of factoids but otherwise seem limited in their overall comprehension. Our solution has been to keep a structured laboratory format, in which students experience a directed and structured hands-on approach. We have, of course, experienced going too far even in this direction. As an early test site for System 7, we instituted features such as Publish and Subscribe to automatically provide structure for labs and funnel students' data to analysis software. The consequence was that the students lost insight into the process of data collection and analysis. The solution was to eschew certain technological capabilities and concentrate on our pedagogical goals. For example, data are tabulated and analyzed by hand.

Marketing

I have not resolved the problem of my deeply ingrained academic training that makes it awkward for me to be engaged in what appears to be self-serving behavior. To some extent, I am old enough so that when I am presenting MacLaboratory software, I don't mind seeming a bit of a salesman and I'm not worried about tenure. At least I use and believe in the product. What is harder is to deal with the erroneous claims of incompetent competitors, or worse, the well-meaning but flawed attempts of shareware or freeware distributors. For example, MacLaboratory's Reaction Time is the most stable, complete, and accurate program for millisecond-based response timing and screen displays that we have tested. But how do we alert the researcher who is making a decision? Do we expect researchers to be sufficiently hardware knowledgeable to test the accuracy of any software purchases? Should we advertise? Is advertising a tolerable behavior for an academic? I would, at the least, propose that any software on any platform that claims millisecond accuracy, including those that have dedicated hardware timers, must present a screen-by-screen timing test for validation. We present such a timing accuracy test, and, as one would predict, an image cannot be presented faster than the screen refresh rate. In our case, we can present a small image in 24-bit color within the single refresh rate for most Macintosh monitors. In field testing, there is wide variability across machines. Beyond the obvious hardware considerations of different models, most users do not seem to realize that software configurations and system versions and extensions have a major impact even on otherwise identical models. Not even identical machines are the same! Well-written experiment-generating software usually determines and adjusts for the machine model,

operating system, monitor configurations, and so forth, but it cannot anticipate all possible user set-ups. In the Macintosh, for instance, most common screen saver programs and network environments are not compatible with consistent millisecond accuracy. The only solution is to provide tests of "true time" for the particular machine. The implications for the discipline seem to be that a level of technological sophistication needs to be as much a part of graduate training as research design, and that certain software performance standards need to be agreed upon.

Teaching Outcomes

In our experience, failed outcomes required the computer to serve as a tutor. Passable outcomes required interaction with information. Successful outcomes required the act of creation. This is clearly a technological variation of the old saw that to hear is to forget, to see is to remember, and to do is to understand. This summary of our outcome data seems to be true for both student evaluation and content mastery measures that we have taken over the years.

Some of the most powerful educational effects have come not from disciplinary software, but rather from productivity software such as graphics, word processors, and spreadsheets. These tools promote the act of creation. Pursuing this concept of creating to learn, I have been requiring that HyperCard or MacLaboratory software be used as the medium with which students create their research papers and present and distribute them to their classmates. This is a process approach I term *media mentoring*. If you measure exam and report performance of students getting the same course but without the software, you find that attractive, popular, interactive, graphic, and colorful tutorial software appears to contribute little to a student's knowledge. However, tools and simulations that create unique experiments do contribute significantly to students' knowledge as measured by performance on content criteria. It may seem obvious to say—although this is seldom tested—that a required outcome for academic computing must be improved knowledge.

Another desirable outcome might be student affect as measured by traditional student evaluations. In my experience with MacLaboratory for Psychology, an "accidental" experiment (Chute, 1990a) has allowed me to compare data from the same evaluation instrument across the 15 years when the curriculum was taught without computers up to 1984, with data from the 8 years of teaching with computers since. Because this curriculum has many successful components that have been constant since its early days when it was pioneered by Robert S. Daniel, it is possible to make a comparison of the effects of computerization per se. The "experiment" is of course flawed by the passage of time, the mix of different universities, possible Hawthorne effects, and so forth.

Nonetheless, as can be seen in Figure 1, there is a consistent and significant improvement for each of 18 measures in the computerized version of the course. In this case, students are evaluating the instructor in the course.

Figure 1. Student evaluation data and factor analysis results comparing a similar curriculum before and after computerization show a consistent superiority of the computerized version of an introductory psychology laboratory (Chute, 1990a; Chute & Daniel, 1990a).

About 10,000 student evaluations make up these data. A factor analysis done on a couple of occasions has consistently yielded three factors we call *mastery*, *structure*, and *motivation*. Mastery indicates how well the teacher and students seem to know the material. Structure indicates how well the course is integrated and satisfies learning objectives, and motivation is related to the quality of the student/teacher interaction. All measures and factors showed improvement in the computerized version of the course. Even though the large sample sizes make statistical significance easy to achieve, the caveats already noted make any conclusions regarding the effects of computerization tentative. It was surprising that the largest single improvement was in class discussion value (loading on the motivation factor), particularly if we recall the anguished speculation of a few years ago that computers would depersonalize the classroom and inhibit student/teacher interaction. Gratifying was the improvement in delivering the major objectives of the course (mastery), which were to provide students with a grounding in the logic of causation, the fundamentals of experimental de-

sign, the principles of data analysis, the process of original research, and the presentation of their work. One student described the difference between the psychology laboratory and other lab courses thus: "In physics you drop the same two cannonballs that everyone has dropped for the last 300 years since Galileo, but in MacLaboratory for Psychology you design and conduct an experiment of your own that may never have been done before." This act of creation on the part of students is enabled by the concept of the computer as a virtual instrument on a disk, an emergent property of creating tools instead of tutors in courseware development. The third and fourth largest improvements were in the written project report (structure) and in knowledge of subject matter (mastery).

The student evaluation instrument also surveys opinions on the value of specific activities and projects that make up the introductory curriculum. As the discipline has evolved, some topics have come and gone as the zeitgeist dictates. Some have even gone full circle and reappeared! Without meaning to generalize beyond our classroom experiences, I feel comfortable in distilling the

following summary comparing the computerized and non-computerized data. The projects that involve original student research or creative work are popular and facilitate learning; simulations (e.g., operant conditioning of an electronic rat) are less popular than the real thing but students learn more; tutorials (e.g., exploring the human brain) are popular, although students don't learn much.

Some might argue that outcome data is not necessary to support the academic computing exercise. Like film, video, or laboratories, it is simply another technique in the professor's pedagogical armamentarium that works for some people some of the time. Why doesn't academia seem as concerned about measuring outcomes from video watching as it does from computer use? If it weren't for the incredible costs, and the problems of complexity, maintenance, and implementation that currently exist, I might be inclined to take such a minimalist view. I persist with the computer in the curriculum because it seems

to provide mastery of important content that is otherwise impractical to teach. Some hypermedia and media mentoring concepts used in a student's creativity tool kit have shown some remarkable accomplishments that mark the start of a new age in publishing (Noon, 1989) and in the classroom (Doyle, 1990). The concept is useful in "soft" psychology—as, for example, in teaching a "laboratory" in abnormal psychology. In such a context, it is important to convey both the subtleties and the difficulties of differential diagnosis, distinguishing, for example, the transvestite from the transsexual or the manic from the psychotic. But students in the past could not get any hands-on experience, for ethical reasons, and alternatives like video tapes were linear and unidimensional or at best something passively watched. For our multimedia solutions, we use QuickTime movie case studies, in which individual students "meet" people with various psychological conditions. Students can individually explore the

Figure 2. Media mentoring is a process approach that seems useful in skill and concept acquisition; we have employed it in courses on experimental and abnormal psychology, and in neuroscience courses. Students use multimedia to create their own "products" and to explore sophisticated content structures. This screen is from MacLaboratory's Psychological Disorders stacks, which enable students to use DSM-III-R decision trees to interpret or diagnose various clinical cases. The illustrated QuickTime video clip is derived from the work of Alan A. Dahms and James H. Scully. Copyright 1992 by Prentice-Hall, Inc., and licensed to MacLaboratory, Inc.

professional-grade diagnostic criteria and decision-making systems embedded in the software. Figure 2 shows a screen from the Psychological Disorders package.

Academic Credit

The effort required to produce software or to implement it in the classroom has created a good deal of tension and consternation regarding the academic rewards of tenure and promotion. Academics who have worked long hours on code or classroom implementation feel they deserve special recognition for their efforts. However, by analogy, the researcher who develops a new technique or instrument only receives credit for the new discoveries made with it, rarely for the technique itself. In teaching, the academic is expected to do the best possible for students as a matter of course. Although using computers is perhaps more arduous than preparing a scintillating lecture, I am not sure that computer implementation per se, even if accompanied by positive outcomes, deserves any more credit. I am now inclined to argue against any special consideration for academic software development or use unless it results in a "software publication"—that is, some type of peer review as in an EDUCOM Award (Kozma & Vinik, 1990), or a paper in *Behavior Research Methods, Instruments, & Computers*, *Teaching of Psychology*, another similar journal, or some other reviewed distribution system such as Intellimation. Of the software developed at Drexel, the survivors fall into the software publication category; the rest are not used for the most part. Programs that did not survive cost approximately two million dollars in University resource allocation and probably did not justify the expense. It does not seem unreasonable to ask that software publication be a minimum standard for entry into the line of academic credit. In part, my change of heart in this matter derives from repetitive and wasted efforts due to lack of a publication record and history of what has been done in the past; we have been destined to repeat ourselves in technological applications, which wastes ever more scarce resources in advancing psychological applications.

We often think of teaching as a spin-off or trickle down from research efforts. An interesting effect of computer applications developed for our teaching needs was a "spin-up" to research use. My current research involves rehabilitation following brain injury and technological applications for the elderly. A powerful and simple concept is to consider the computer as a brain prosthesis for such people. In the past, computers have been used as cognitive "therapists" (as in tutorial software) without much success with the neurologically compromised patient. The idea behind SpeechWare (Chute, Conn, Dipasquale, & Hoag, 1988) is to customize software to each unique patient's needs. Like the metal hook of an artificial hand, it does not need to exactly duplicate the disabled function to be useful. In this case, SpeechWare is a "spin-up" from our teaching software experience, offering digitized and synthesized speech augmentation, as well as management of household circuitry and telecommunications, for the cognitively and motorically impaired person. Its object-

oriented quality makes it reasonably customizable by students and health care professionals working in this field. The clinical research project represented by SpeechWare has been fed back into teaching in an undergraduate course on interface design as well as in more traditional courses such as rehabilitation psychology.

Conclusions

Neither individuals, nor universities, nor commercial operations can supply the technological sophistication that psychology will need for its research, teaching, and clinical endeavors in the future. The discipline, and we as its members, will have to coordinate broadly inclusive collaborative and cumulative efforts.

REFERENCES

- CHUTE, D. L. (1986). MacLaboratory for Psychology: General experimental psychology with Apple's Macintosh. *Behavior Research Methods, Instruments, & Computers*, **18**, 205-209.
- CHUTE, D. L. (1987). Glimpses of the darker side of academic computing. *Academic Computing*, **2**(1), 40-59.
- CHUTE, D. L. (1989). *Software development: Problems and solutions for the publishing industry*. Invited address to the meeting of the American Association of Publishers, New York.
- CHUTE, D. L. (1990a). Academic angst. *Academic Computing*, **4**(8), 10-14.
- CHUTE, D. L. (1990b). *MacLaboratory for Psychology 2.0 software documentation*. Devon, PA: MacLaboratory, Inc.
- CHUTE, D. L. (1991, Fall). Software piracy in academia. *Psychology Software News*, pp. 72-74.
- CHUTE, D. L., CONN, G., DIPASQUALE, M. C., & HOAG, M. (1988). ProsthesisWare: A new class of software supporting the activities of daily living. *Neuropsychology*, **2**, 41-57.
- CHUTE, D. L., & DANIEL, R. S. (1990a). *MacLaboratory for Psychology instructors' notes* (2nd ed.). New York: McGraw-Hill.
- CHUTE, D. L., & DANIEL, R. S. (1990b). *MacLaboratory for Psychology laboratory manual* (2nd ed.). New York: McGraw-Hill.
- CHUTE, D. L., GAEMAN, D., & ZIEGLER, T. (1987). MacLaboratory Controller: A switch and A/D interface between Apple's Macintosh and peripheral apparatus. *Behavior Research Methods, Instruments, & Computers*, **19**, 205-209.
- DANIEL, R. S., & CHUTE, D. L. (1972). *Student's manual for general experimental psychology* (2nd ed.). Columbia, MO: Lucas Brothers Publishers.
- DOYLE, D. (1990, January). *Emerging technologies: Then, now, and soon*. Paper presented at Macademia Southeast 90, Daytona Beach, FL.
- HEWETT, T. T., & PERKEY, D. J. (1984). The mythical "mountain" of software. *Collegiate Microcomputer*, **2**, 207-210.
- HUFF, C. (1992, November). *Network resources for psychology*. Paper presented at the meeting of the Society for Computers in Psychology, St. Louis.
- KOZMA, R. B., & VINIK, E. (1990). Can software publishing keep faculty from perishing? *Syllabus*, **9**, 10-11.
- NOON, J. P. (1989). *Emerging technologies in the higher education marketplace*. Keynote address for the meeting of the Association of American Publishers, New York.
- ONARAL, B. (1990, January). Engineering courseware: A developer's perspective. *Academic Computing*, **4**(6), 12-14.
- PERKEY, M. N. (1986). The effect of a machine-rich environment on courseware development: The process and the product. *Behavior Research Methods, Instruments, & Computers*, **18**, 196-204.
- WESTALL, R. [F.], PERKEY, M. N., & CHUTE, D. L. (1986). Accurate millisecond timing on Apple's Macintosh using Drexel's MilliTimer. *Behavior Research Methods, Instruments, & Computers*, **18**, 307-311.
- WESTALL, R. F., PERKEY, M. N., & CHUTE, D. L. (1989). Millisecond timing on the Apple Macintosh: Updating Drexel's MilliTimer. *Behavior Research Methods, Instruments, & Computers*, **21**, 540-547.