An Introductory Course on M Worth Exporting

by Richard F. Walters

Introduction

M is growing in popularity in many application fields, but it has not yet achieved widespread recognition in universities and colleges. The few campuses where it has been introduced are those with faculty or staff long familiar with the language and aware of its benefits. The number of new "converts" in academia to the value of the language is indeed few.

We could postulate many reasons for the relative lack of acceptance of M in an academic setting. Without question, the limited availability of M prior to the proliferation of personal computers was a factor. M required dedicated minicomputers in most implementations and this alone was a major deterrent to academic settings where it enjoyed, at best, a secondary role. Even when it became more widely and conveniently available, many thought that since the M language was already a decade or two old its features might be outdated. Perhaps even the intemperate rush to the relational model led many computer scientists to ignore M because it represented the old hierarchical structure.

M has many features that make it ideal as an introductory language for people with no background in computers. A recent experiment to use M in introductory computing classes at the University of California, Davis, has become quite successful, so much so, that we think it may serve as a model for courses elsewhere. This article summarizes the basis on which that course was developed, to describe its components and the role M plays in the course, and to show how this experience might be extended to larger enrollments and other campuses.

Background

Introductory computer courses tend to focus on one or more of three general areas:

- Knowledge of how computers work, including some historical perspective;
- "How to" courses on the use of computers in various fields (word processing, spread sheets, databases, etc); and
- Introduction to computer programming.

Available textbooks reflect the orientation of their authors in these respects. Many designers of introductory courses emphasize a need for hands-on training. Patterson claims that an introductory course must have hands-on elements including programming in order to be successful, as exemplified by the textbook he and his colleagues have published.[1] Other introductory courses focus on historical perspective with a lesser focus on applications.[2,3] Finally, some courses focus almost exclusively on programming and pay minimal attention to the history of computing and no treatment of computer applications such as word processing.[4]

On the Davis campus, all three types of introductory computer classes exist. The College of Agricultural and Environmental Sciences, the "Applications of Microcomputers in Agriculture" course introduces students to applications, but does not cover programming or an extensive review of computer history and components.[5] The Department of Computer Science offers two courses. The first is an introductory programming course for nonmajors based on Pascal. The second course, described in this article, covers all three basic concepts (historical perspective, applications, and introduction to programming).

There is still room for a course that meets all the needs of an introductory computer course as outlined earlier and at the same time presents M as a language of choice for such a course. There is a course that introduces computers to nonmajors, ECS 15. This article traces the development of the course and describes the general format and content, including the use of M. It also presents some of the teaching techniques used in presenting the course and concludes with some suggestions on how to expand the course for wider use at Davis and elsewhere.

Course Design for ECS 15

The University of California requires all undergraduate majors to take a number of general education courses outside their major area of interest. Courses approved for this desig-

nation must incorporate a number of features, one of which is a term paper. ECS 15 meets those requirements, while serving its principal purpose to demystify the computer.

The goals of ECS 15: "Introduction to Computers" include all three general areas described earlier. The decision was made to exclude discussion of database systems from the course because it would require more in-depth background than could be presented in a ten-week academic quarter. Instead, a second, upper-division (junior-senior level) general education course on database systems will be presented for the first time during the 1994 winter quarter.

The inclusion of programming to the extent that students can understand code written in M might be considered controversial. It seems, however, that individuals who understand how programs are written, even in a limited way, will be more effective in communicating with programmers in future dealings.

The course is a four-unit class, with three lectures and one three-hour scheduled laboratory each week. The lectures present an introduction to computer components (hardware and software), typical applications of computers, concepts of programming, and the impact of computing on society. A set of lecture notes covering the material presented in the core lectures is available for student access; students can print the notes as soon as they gain experience in using a word-processing package taught in the laboratory exercises.

There are nine scheduled laboratories during the ten-week academic term. All sessions are completed using personal computers with MS-DOS in a computer classroom with thirty networked PS/2 systems. There is a comprehensive laboratory manual to accompany the scheduled exercises, covering the following topics:

- Introduction to DOS (one laboratory);
- Introduction to WordPerfect (two laboratories);
- Introduction to EXCEL (two laboratories); and
- Introduction to Programming in M (four laboratories).

The inclusion of programming to the extent that students can understand code written in M might be considered controversial. It seems, however, that individuals who understand how programs are written, even in a limited way, will be more effective in communicating with programmers in future dealings.

In addition to the lectures and laboratories, the course requires a 2,500-word term paper dealing with some aspect of the application of computers in their major field or some other application domain. This assignment should help students improve their writing skills, so there is a heavy emphasis on preparation, organization, and presentation of the referenced material. Students are required to submit three separate items:

- A prospectus, about two weeks into the academic quarter, in which they describe the topic they plan to research and their initial evaluation of the availability of material;
- A progress report, due the fifth week of class, in which they describe in greater detail the questions to be answered and references located to date; and
- The final term paper, due two weeks before the end of the term.

The prospectus and progress report are evaluated and returned in a timely manner to facilitate writing the final paper, which is also graded and returned in time to allow the student to make corrections and resubmit to recover up to half of the points lost in the first submission of the paper.

A midterm and a final exam test the student on concepts introduced in the lecture and laboratory. Weighting of the separate components of the course is as follows: laboratories, 40 percent; term paper, 25 percent; midterm, 15 percent; and final, 20 percent. Students can also get extra credit for doing advanced work in laboratories and for submitting additional information on course-related topics during the quarter.

The laboratory manual and class lectures have been improved with minor changes as suggested by students and teaching staff during the several course offerings.

The text used in the class, in addition to the lecture notes and laboratory manual referenced earlier, is *The ABCs of MUMPS*.[6]

Use of M as an Introductory Programming Language

Four laboratory exercises in the course are devoted to learning basic programming concepts. An early version of UCD MicroMUMPS is used in conjunction with a shareware editor (Q edit) to allow the student maximum flexibility in moving between the editing and program execution environments. The first two laboratory exercises take students through sim-

ple commands and the concepts of local and global variables and arrays, and they introduce the principal string-handling functions in M. The third laboratory introduces the idea of editing programs following a scripted scenario. The final laboratory presents a problem requiring the students to design and implement their own M routine, using $ORDER and other more advanced features to demonstrate the understanding of execution flow control and manipulation of M hierarchical arrays. By the end of these four laboratory exercises, students have a good grasp of the fundamental actions that a computer performs, and they learn the importance of manipulating nonnumeric data. This knowledge is reinforced by the final examination, which includes a program in M, handed out one week before the examination, about which students must answer questions on execution flow, the purpose of different commands, and elements of the syntax of the program.

The overall course design appears to meet the underlying objective of demystifying the computer for students who may have had some major reservations about computers in general.

Our experience has shown M to be a highly useful language for teaching fundamental concepts of programming. The success of student performance in the final laboratory and in answering questions about the program included as part of the final exam demonstrates that these concepts have been mastered by the vast majority of the students. They accept the value of computers in processing nonnumeric data, the type most of them are likely to work with in their respective major fields. The materials prepared for this course have served as a valuable resource for others wishing to learn M. Outside educators have requested and received copies of the laboratory manual and the associated UCD MicroMUMPS code with specific routines as well as the Q edit program adapted to permit students to toggle back and forth between the program and editing environments. Individuals using this material independently have reported their ability to learn M quickly and conveniently without requiring hands-on personal assistance. Students using this independent approach have ranged from computer science majors to programming novices.

The overall course design appears to meet the underlying objective of demystifying the computer for students who may have had some major reservations about computers in general.

M is used by two of the three instructors who have taught this class since its inception in 1991. The third instructor uses SCHEME, a LISP dialect to introduce programming concepts.[7] Both seem to be accepted by students, but no detailed comparison has been made of the two methods.

Pedagogic Techniques

Some unusual techniques of this course appear to have aided its success. The instructor makes an effort to get to know each student by name, taking individual photographs at the start of each academic quarter. Electronic mail provides rapid feedback on student questions and comments. It also serves as the principal means of communication for announcements and general information. Students readily accept and appreciate the quick response afforded by this means of communication.

The instructor has also adopted a technique for rapid feedback that was recommended by a national study of teaching effectiveness.[8] At the end of each lecture, students answer on paper two questions: "What is the most important point you learned in this class?" and "What is the most important unanswered question at the end of this class?" These comments are collated, summarized, and reviewed with students at the start of the next lecture.

Using a technique proposed by Patterson, the laboratory exercises each include a pretest, which must be completed and submitted in order for the student to enter the laboratory classroom, and a post test, with questions to be answered during the student's progress through the exercise.[9] This approach ensures that the student has prepared adequately for the laboratory exercise, and it offers a check on the knowledge gained in the laboratory. Students earn full credit for each laboratory successfully completed (as indicated by a satisfactory post test); hence, there is no competitive stress to outweigh the learning objectives of the laboratory exercise.

One especially vital technique is that none of the teaching staff touches the students' keyboards. This removes the tendency for the instructor to "solve" the students' problems. Instead, the student is guided through steps to solve or correct the situation that led to a question.

The interim steps required to complete the term-paper assignment were designed both to intervene early in the term-paper writing process and also to provide incremental guidance in effective preparation and submission of the report. Perhaps the most important component is giving the student an opportunity to make corrections on the final paper.

Examinations are deliberately designed to be learning experiences rather than threatening or recitative ones. Practice tests are reviewed shortly before each real one is given; there are no multiple choice questions, only those requiring short answers. Questions give students an opportunity to think about new concepts for which course material has provided background for analysis.

Despite use of electronic mail, the instructional staff holds regular office hours. In addition, weekly sessions are held with the instructional staff to review material for the coming week and note revisions required from the week just completed.

Grading is not done on a curve. Students are informed that they are not competing against each other, but only against themselves. Opportunities are given for extra credit, but scoring is generally matched closely to an absolute percentage score: A equals 90 percent or better; B equals 80 percent to 89 percent; and so on.

These techniques have helped make the course personal and individualized despite large enrollments. Students respond positively to the individualized attention and contribute willingly by providing constructive feedback.

Evaluation

To date, the response to the course has been overwhelmingly positive. Numeric evaluations have averaged approximately nine on a scale of ten. Students reported that they have learned, and they appreciate the extra effort put forth by the instructional staff. They recognize that the course covers a large amount of material, but almost uniformly they agree that the course helped demystify the computer. Their reaction to the term paper depends on prior background, but even most upper-class students report having learned new techniques for term-paper writing.

Students also report that, though they do not feel confident about writing programs themselves, the course has aided this process to the point that they would not feel intimidated by talking to a programmer about a proposed software problem.

Another indirect measure of success is that many students recommend the course to their peers. Some students have requested that it replace other required introductory computer courses in their academic major. Several former students have volunteered to assist in the instruction of the course. Almost all students agree that a course of this type should be required for graduation from college.

In short, the course seems to be meeting its goals in providing an effective learning experience for the students able to enroll

in it. The largest unmet need, however, is that of meeting the enrollment demand. This problem is discussed in the next section.

New Modes of Learning

The success of this class has led to increased demand beyond the capabilities of current instructional resources. This demand is expected to continue to increase. In fact, if one accepts the notion that some form of introduction to computers is a necessary component of a college education, then the demand on this campus is likely to increase dramatically. Of the roughly four thousand entering freshmen each year, current estimates are that perhaps half of those students may not be getting instruction in use of computers. This estimate alone would suggest a need to expand enrollment to accommodate the five hundred students or more per academic quarter.

In light of this pressure to expand the offering, we have begun to explore ways in which this course might be offered to a larger number of students in the future. New methods must be found to present the materials to students, using different means and different approaches to scheduling. These needs are by no means specific to this course. Many techniques proposed are widely applicable to other courses.

The present format of the course (lectures, laboratory exercises, a term paper, and two examinations) is labor intensive in the specific areas of laboratory supervision and term-paper guidance and review. The course requirements are resource intensive in that they require scheduling of laboratories. Technical equipment is used in lectures (to demonstrate concepts, illustrate computer applications, introduce laboratory exercises, etc.), requiring a classroom equipped with network and computer-projection equipment. If the size of the class were expanded without changing its format, the number of suitably equipped lecture halls would become a limiting factor.

Another resource that might become restrictive is access to the computer network. This is a matter of balancing the number of lines available and the number of computers that can accommodate instructional applications.

We have analyzed these potential bottlenecks to expansion, and after some reflection, we have concluded that the following six approaches need to be taken.

First, videotaping lectures and demonstrations is essential. We nearly have completed a series of videotapes of many of the lectures in this course. Some already have been used in place of lectures (though the instructor remains in the classroom). We have found that the material presented by scripted

videotapes rather than by live lectures is better, partly because of the ability to incorporate special material and graphical effects. In addition, students can view tapes at times convenient for their schedules. Additionally, having the material on tape permits more effective review of the material. We are working to make the tapes more broadly available, including checking out tapes, nighttime use of educational channels to broadcast the lectures on local cable television, and more sophisticated online remote retrieval of the presentations.

Second, such a course needs more effective mechanisms for the use of electronic mail and other network resources for course management and exchange of information. This is a broad topic that incorporates several smaller components, including distributing class information, providing more "WYSIWYG" (what you see is what you get) presentation of problems encountered by students or graphical information supplied by instructors, and developing suitable electronic forms for submitting class assignments.

Third, the facility to give interactive, online assistance from a remote location is central to being able to do away with scheduled laboratories. If a student is able to receive online help while doing an exercise at home, then one tutor or teaching assistant might be able to serve the needs of a much larger class enrollment.

Fourth, more effective administrative solutions to review and report on term-paper and other comparably long written assignments need to be developed. Additionally, if all such assignments could be processed entirely electronically, rather than requiring hard copy, it would be very useful.

Fifth, we need to package instructional material (laboratory exercises, lecture notes, and other material) in forms that can be accessed remotely without loss of quality of the material in comparison with hard-copy originals.

Finally, capitalizing and expanding on certification and evaluation procedures that can be done remotely via computer networks would round out the introductory course.

Work has begun on several of these fronts. Videotapes have already been made introducing all the laboratories and presenting material from approximately half the lectures. Extending the flexibility of the computer network is beginning. There are many components of this project. We expect by fall 1994 to be able to eliminate some or most scheduled laboratories, allowing students who have their own computers to complete the assignments outside the classroom.

Computer packages providing remote tutorial assistance have appeared on the market, such as Timbuktu, marketed by Farallon Computers.[10] Although none of these systems meet the complete requirements of remote tutorial assistance, they represent a good start, proving that the technology is capable of resolving some previously intractable technical problems, such as linking heterogeneous networks to provide effective, seamless computing to be carried out across such networks.

Work has also been done on improving the flexibility of electronic mail, so that nontextual information (graphics, various fonts, and font sizes, etc.) can be incorporated in such messages.[11]

These and related recent developments suggest that the distance-learning goals of this project are all technically achievable, though work remains to be done to form a coherent framework for the different program elements suggested in this discussion.

Discussion: The Role of M in Introductory Computing

M plays a pivotal role in the introductory computer class described here. There are some limitations to the use of M that bear review, however.

The various items produced in conjunction with this project are valuable tools not only for teaching introductory computer science, but also for helping people learn about the M language (even if they are already computer literate). These include the laboratory manuals and videotapes thus far completed. The videotapes on programming in M offer some new approaches to presenting concepts such as variables, arrays, and text processing.

These materials certainly could be improved and a better version of M should be used in the course. Such a version would have to be easy to use and should include all current language features. UCD MicroMUMPS is acceptable as a start, primarily because of its effective error messages and because a shareware editor has been linked to it to provide a seamless editing/programming user environment.

Unfortunately, no commercial system currently available provides these key features. Whereas vendor-supplied editors may be acceptable to the experienced programmer, they present a formidable challenge to the naive user, who is struggling with programming concepts and does not need the extra baggage of learning about separate editors, and compile and run commands, etc. Further, the error messages available are often uninformative, partly because they fail to identify the exact locus of the error as encountered by the M interpreter.

The experience with M proves that this language is better suited to teach novice computer users about programming

concepts. Other languages are unlikely to provide a comparable mix of features so necessary in the naive learning environment. Hence, M has an opportunity to provide a valuable service to a large population of students who are in dire need of more knowledge about computers. What remains is to find ways that they can be reached. We can offer help in terms of providing a useful package for learning, but it remains for the M community to identify institutions that might be interested in accepting and making use of a tested package of this type.

Conclusions

There is a growing demand for introductory instruction about computers, including how they work, how to use certain key applications, and how they can be instructed to perform simple tasks. This Davis campus course has been successful in meeting this need. Because of a local requirement to extend the class to a much larger audience, independent study aids are being developed to augment the formal lecture and scheduled laboratory method currently used to present the course. These materials could form the basis of courses taught elsewhere, both in scheduled and independent study mode.

I am interested in working with individuals or institutions seeking to take advantage of these materials. The M community might stand to benefit in a number of ways, not the least of which might be a significantly heightened visibility of the M language.

While much remains to be done, this project has shown that M is a highly effective tool for teaching principles of computers at the introductory level. Those of us involved hope that this small start can serve as a stimulus to the initiation of more projects of the same type. **M**

Christopher Reid, a Davis student, has given invaluable assistance in preparing the laboratory manual, the early televised experiment, and videotaping. Drs. Sergio Alvarado and Vincent Barone also have taught the course. Andrew Austin, Jon Barbour, Paul Chang, and Dorothy Yee also contributed to the course materials.

Additional support for the work described in this article was provided by the University of California, Davis, its Undergraduate Instructional Improvement Fund, Adobe Computers, and Farallon Technologies, Inc.

Dick Walters is a professor of computer science at the University of California, Davis, and designed this class. He is an active MTA member and executive editor of *M Computing*.

Endnotes

1. D.A. Patterson, D.S. Kizer, and D.N. Smith, *Computing Unbound* (New York: W.W. Norton and Co., 1989).
2. H.G. Kershner, *Introduction to Computer Literacy* (Lexington, Massachusetts: D.C. Heath and Co., 1990).
3. H.L. Capron, *Essentials of Computing* (Redwood City, California: The Benjamin/Cummings Publishing Co., 1992).
4. W. Savitch, *Pascal: An Introduction to the Art and Science of Programming*, 3rd ed. (Redwood City, California: The Benjamin/Cummings Publishing Co., 1992).
5. R.E. Plant, *Course Pack: Applications of Microcomputers in Agriculture* (Davis, California: Department of Agricultural Science and Management, University of California, 1993).
6. R.F. Walters, *The ABCs of MUMPS* (Bedford, Massachusetts: Digital Press, 1989).
7. Lightship Software, *MacScheme Manual and Software* (San Francisco: Scientific Press, 1990).
8. R.J. Light, "The Harvard Assessment Seminars: Explorations with Students and Faculty about Teaching, Learning, and Student Life," second report, Harvard University Graduate School of Education and Kennedy School of Government, Cambridge, Massachusetts, 1992.
9. Patterson.
10. Farallon Computers, *Timbuktu for Windows (User's Manual)* (Alameda, California: Farallon Computers, Inc., 1993).
11. Adobe Systems, *Adobe Acrobat Starter Kit* (Mountain View, California: Adobe Systems, Inc., 1993).