M COMPUTING

Executive Editor
Richard F. Walters, Ph.D.

Managing Editor
Pamela McIntyre

Editorial Board
Richard G. Davis, Ph.D.
Gordon E. Gebert
Michael A. Ginsburg

Review Board
Mark V. Berryman
Robert S. Craig
A. Clayton Curtis, M.D.
Richard G. Davis, Ph.D.
Ruth F. Dayhoff, M.D.
Michael A. Ginsburg
William J. Harvey, Ph.D.
Frederick L. Holtz, Ph.D.
Arthur Lee
William J. Majurski
David Marcus
Frederick D.S. Marshall
Thomas Munnecke
Helmuth F. Orthner, Ph.D.
Roger Partridge
Gail Penrod
Thomas C. Salander
Kate M. Schell
Hans T. von Blanckensee, Ph.D.
Richard F. Walters, Ph.D.

M Computing is published by MUMPS Users' Group trading as M Technology Association, Suite 205, 1738 Elton Road, Silver Spring, Maryland 20903. Phone: 301-431-4070, Fax: 301-431-0017.

M Computing (ISSN 1060-7684) is copyrighted by the M Technology Association. Material in this magazine may be excerpted by reviewers providing that credit for the original publication is made to *M Computing* (specify volume and issue numbers and date), a publication of the M Technology Association, Silver Spring, MD 20903 (Phone: 301-431-4070). Permission for reprinting articles from *M Computing* must be granted by the editor.

The information in this publication is believed to be accurate as of its publication date. MTA is not responsible for inadvertent errors. All trade and product names referenced are the service marks or trademarks of their respective companies. Opinions expressed are those of the individual authors.

M as Part of a Hybrid Solution

Richard F. Walters

Richard F. Walters

The editorials that appear in *M Computing* usually emphasize recent events in the M world: the effects of the MTA conference in its first year merged with DataBase & Client/Server World, actions affecting the new standard, and often feature articles in the current issue. I have attempted to back away from the immediate concerns of those events and relate them to a larger world. I would like to continue with that theme this time, focusing on a recent experience here at UC Davis.

As many of you know, I teach database systems to computer majors, offering a two-term (10 week academic "quarters") sequence that begins with file structures, moves through "structural" models (hierarchical, with a passing reference to network) to the relational model, then on to semantic models (Entity-Relationship, SIM) and on to Object-Oriented database systems. Students get an opportunity to program several projects, beginning with raw code implementing direct file access using hashing (soundex-based) techniques, moving through two relational packages (Access and Oracle), an ER package (ZIM), an advanced hierarchical system (M, of course), an OODBMS (POET), and concluding with individual projects of the students' own choosing (subject to instructor approval). The M assignment gives students a real-world problem I faced a few years ago: a list of committee memberships, complete with typos, listed by first initials followed by last name which must be inverted into a list sorted by each individual's last name, edited, with some output requirements to liven the exercise.

The students taking this course (about 100 in the first term, 50 who stick it out for both terms) are hard core computer majors, familiar with C, C++, UNIX, and in most cases owning their own PCs usually running LINUX. They have about ten days to complete the M project, and about three weeks for the final project. I offer 3 points (on 100 points for course) extra credit for anyone who can program the M assignment in C or C++. About five this year completed the C version, two of whom completely matched the output requirements easily met with M code. They concluded that it was a great deal harder in C or C++ (surprise, surprise).

I give a take-home final in the second quarter, and this year I included a question that asked the students, given the exposure to different models and systems, how would they go about designing a new database application from scratch (i.e., the database did not exist when they started the project).

Some interesting results emerged from this year's class. First, three teams of students selected M-based systems for their term projects: one took on a beta test version of ESI Objects, and two used somewhat more conventional M packages to develop interesting applications intended to serve as assignments for future classes. Their solutions were interesting and innovative (even though they missed some of the niceties of efficient global file design). The ESI Objects assignment will probably be adapted for class use next year.

Second, of the nearly 50 answers to the "pick your system" question, more than half included M as a component of that answer. Not stand-alone M: that doesn't fit with their backgrounds; M as a partner in a multi-language solution. Citing the basic strengths of persistent data, string handling, and an effective hierarchical sparse array file structure (well, they didn't use quite those terms), these hard-core computer scientists were convinced that M should be a part of a well-designed solution to database applications.

What lessons can we derive from this experience? Several, in my opinion. First: offered a chance to get a fair exposure to M, a level-headed computer scientist trained in other disciplines cannot help but recognize the obvious strengths of M cited in the preceding paragraph. Second: their designs in almost all cases capitalized on these strengths in a hybrid environment; they did not go whole hog to M solutions, but proposed using it in conjunction with other languages for aspects of a DBMS solution which they felt were better handled in other languages: GUI front ends, system primitives, and other areas where their familiarity with C or another solution seemed to offer better options.

It may not be a coincidence, but one of the guest speakers in the second term described a hybrid solution in which M played a part. This gentleman, a seasoned veteran M programmer, had reached a similar conclusion and showed that it would work (though his approach was considerably more sophisticated than any proposed by the students in my class). There was a gap of two weeks or more between that guest lecture and the time I handed out the exam, and nothing in the wording of the question suggested any reference to that or other guest lecture. I think that the logic of this approach was something they each thought out for themselves.

These students are the programmers and managers of tomorrow. They go into the world with a pretty good understanding of what's available, and they see M as an attractive partner in a multi-language world of the future.

Maybe we can draw some conclusions from this class. Maybe we should be emphasizing M as a partner. Maybe we should be enriching the ways in which we link M to other systems, drawing on their strengths just as we do on those of the M with which we are so familiar. Maybe, too, we need to be a little more assertive in suggesting M as a partner to individuals who are less fortunate than us or my database students. After all, if they can be persuaded with such a minimal exposure, mightn't some others who face impossible hurdles imposed by C tools also step back and examine the obvious benefits of M?

I have to conclude these reflections with a non-scientific, anecdotal observation: it seems to me that many of my M friends are beginning to do exactly what I propose, at least in adopting hybrid solutions and in a few cases, proposing M as a partner to other languages.

Wouldn't it be nice if more of us and the outside world did the same? **M**

Richard F. Walters, Ph.D., is a professor at the University of California, Davis, and the executive editor of *M Computing*. Write to him care of MTA's managing editor or e-mail to: Walters@cs.ucdavis.edu

---


Dear Readers,

Beginning with this issue of *M Computing*, we are adding a new feature. For those of you interested in finding additional information on subjects covered within each issue, we will include a bibliography of current references related to the feature articles. For this issue's bibliography on internationalization, see page 39.

Please note that the length of each reference list will vary depending on how much information is found online. At a minimum, we will go back two years from the date of publication.

—The Editor