Grow Your Own M Programmer

by Kate M. Schell

Boston-based M programmers may not believe it, but in some parts of the country, and the world, M programmers are a rarity. I frequently meet managers who wonder where they will find their next M programmer. Sometimes, all they need is a reality check, or a broader look around; there are many good sources for experienced M help.

Before designing a training program for M programmers, here are some tips for finding people who already know M:

- Ask your M programming staffers who they know. (And be sure that they are being compensated at the appropriate rate for their experience!)
- Contact the M Technology Association's Job Referral Service for listings of programmers looking for work.
- Contact your local users group (LUG) to find out if it keeps lists of people who want a new situation. There's a listing of LUGs and their contact people in MTA's current *MSources*.
- Check with the colleges and universities that offer M courses, again, listed in *MSources*.
- Advertise in M publications at both the local and national level.
- Work with the recruiters who specialize in M programmers. These people advertise in M journals and on electronic bulletin boards.

Let's assume that hiring is hopeless; you've tried some or all of the above, and experienced people are not available. It's time to grow your own M programmer(s). I selected the word *grow* deliberately. Growth is a process; so is staff training. The more effort you put into cultivating staff, the more you will get out of each member. Education takes time: time for preparation, time for presentation of the material, and time to review students' progress. (The educators among you will recognize the equivalence of lesson plans, classroom teaching, and correcting homework.) This minimal list of functions does not take into account time to keep the teacher on top of developments in the profession, nor the day-to-day mentoring that often makes the difference between success and failure for a new programmer.

Why invest in training? Why not just toss a new staff member into a sink-or-swim program with a manual or two for guidance? I learned all of the basics in most of my M jobs by scouting out manuals and documentation, pestering people with more experience than mine, and by trial and error. The "If they can't figure this out they don't really belong here" approach is common in the M community. I am convinced that it is a bad idea. I earnestly hope that most of the code I wrote in my first two years has been replaced by now. I am certain that most of us would hate to have to maintain the first five routines we wrote after our discovery of indirection!

Many M programmers in my acquaintance learned to code by emulating existing code in the system. If there is code in your system that you find hard to maintain, make sure that it is not copied. Take the time to tell staff members what is good and acceptable, and you will increase your chances of getting good and acceptable code.

It costs a few thousand dollars to provide basic training for an M programmer. It can cost a lot more than that if you have to terminate someone's employment. Rewriting code that has been produced by staff with an inappropriate coding style is also expensive. You can cut your losses on both ends by spending some time up front on training.

Training Program Basics

How do you set up a training program? The essential steps are to:

- Identify a training coordinator;
- Identify training-program goals; and
- Find the means of achieving the identified goals.

It is important to have a coordinator to oversee the training effort. The key person should have an interest in people and in the success of the people being trained. A knowledge of M programming is, of course, essential; however, the person need not be an expert programmer if there is access to expert programmers who have good communication skills. Somewhere along the line, you are going to need a trainer. If the program coordinator is not going to be the trainer, the coordinator must be able to identify and evaluate the success of those who do the actual training.

The program coordinator should identify the goals for the training program probably in consultation with those who will be directing the work of the newly trained staff. A good training program will focus not only on new staff, but on the continued development of existing staff. What level of training and experience ought to be required of a junior programmer, a senior programmer, a project leader? (If these titles do not fit your organization, plug in the appropriate ones.) There probably will be more than one level of training, hence more than one set of goals; there may be more than one step to the completion of each goal.

Each discrete set of training goals should be explicitly stated in a document. These documents will become the basis for your course development, and for evaluating trainers and trainees. Take the time to do this well. If you cannot state what you want, you probably will not get it. Surely most of you have heard the following at one time or another from someone who "didn't feel the need" for a written goal: "Specification? What specification? I spent ten minutes talking to the users!"

As an example, let's assume that the goal is to enable new staff members to write reports. That goal might be fulfilled through:

- An introduction to M programming;
- An introduction to any tools or utilities in use in your shop;
- An explanation of various devices and device selection;
- An introduction to the program and global documentation for the systems for which reports are required;
- Description of routine and report naming conventions; and
- Presentation of in-house coding standard documents and validation tools. (A panel discussion on this topic will take place at the 1994 MTA Annual Meeting in Reno, Nevada.)

If the goal were to enable a new programmer to do maintenance programming, staff members probably would need all of the five elements identified above, plus:

- Training in diagnostic techniques (What goes to the error trap? What doesn't hit the error trap? When is a problem a software problem versus a hardware problem? Is it really a coding problem, or is the tape drive off-line?);
- Use of backup and restoration procedures;
- Database repair procedures;
- Procedures for documenting solutions;
- Telephone use protocols (How to answer a call—for example, with organization name and personal name; how often to contact customers with open calls; who determines that a call is closed?); and
- Identification of billable versus nonbillable support.

These are just two examples of sets of training goals. You will have to determine the needs of your organization and your staff. In many cases, you will be starting from scratch. The important thing is to get started.

The coordinator does not have to present all of the material required to attain a set of goals, but must be responsible for ensuring that the information is provided and, in most cases, reviewed with the new staff.

Creating goals should include identifying an appropriate time period for demonstrating proficiency in that skill. For example, if you want to be able to produce reports, once a staff member is trained, the individual should be able to code simple reports. Projects ought to be aimed at that level, and should then get progressively more difficult. Of course, it is often difficult to persuade users to specify simple reports when needed. If you can spare the time, you might have new programmers code new versions of existing reports. They can then compare their code to code written by a more experienced programmer. (Note that the code you are comparing it with ought to be worth emulating.) New programmers can also gain valuable experience by rewriting existing M code in structured format, or rewriting to bring existing code in line with the current M standard.

The In-House Expert and Other Resources

Once you have identified goals, identify the resources available to help achieve that goal. Among the resources you ought to consider are: instruction manuals, documentation, trainers and courses, and computer-assisted instruction.

I have two current favorite M manuals, both available through MTA. *The ABCs of MUMPS* is a good teaching manual, and *The Complete MUMPS* is my favorite reference manual. Many of the M vendors supply M manuals and system documentation. (MTA also has a catalog of publications and software available free of charge.)

Documentation available to staff should include:

- User documentation;
- Routine documentation;
- Global documentation;
- Description of in-house tools and utility routines;
- In-house standards for naming routines and globals; and
- In-house programming standards.

Programmers need access to user documentation to understand functionality. It also helps to establish whether a user with a complaint understands the system. Routine and global documentation are a sore point for many companies. Provide what you have available, and establish requirements to produce documentation for new and modified code. Documenting a set of routines before allowing modification can ensure that a new programmer understands the existing code and the proposed changes.

Documenting in-house utilities, programming standards, and naming conventions is another traditional weak spot for M shops. If you have documentation, provide it. If you have undocumented in-house standards, find someone with the knowledge who will make the time to get the information into black and white or bits and bytes.

If you don't have in-house standards, you need to consider establishing some before your code becomes completely impossible to maintain. This is a "pay me now or pay me later" issue. Consultants earn great money fixing code that has been written without coding standards. Do you accept application code that uses `$VIEW`? Do you permit use of `Z` functions and commands? Can application code contain calls to vendor-supplied utilities?

You need at least one person in your organization who is an M enthusiast, and likes to share what he or she knows. This person may have already been identified as the training coordinator. To identify the most likely person, ask people where they take their questions. If that person feels the need for more training before being officially recognized as an expert, identify the training needed, and help him or her to get it. Remember that this person need not be prepared to present classes, but must be willing and able to answer questions, and know where to go for answers otherwise.

You also need people who can give classes for staff. These people could be in-house staff or could be external. There are people and organizations who offer all sorts of courses in M and M-based systems. Among them are professional organizations, colleges and universities, independent training companies, M system vendors, and consultants. Professional meetings, such as MTA's Annual Meeting, offer courses. Independent trainers contract to run courses in offices. Or, if you get together with other organizations in your area, you might consider sponsoring a course at a central location in the vicinity.

Training is one area you do not have to handle alone. Many people train M programmers for a living. For the most part, they have a good understanding of the amount of material that can be absorbed in a day or a week, and they have good experience in presenting that material in an interesting and understandable manner. Some trainers are available to teach on location at your offices. If you are training a number of people at one time, this may be the most cost-effective solution. Otherwise, consider sending people to off-site courses. Sources of outside training include: tutorials at conferences; professional training companies and consultants; universities and colleges; vendor instruction in M programming and tools.

In-house staff members probably are the best-suited to handle issues such as device selection, in-house coding standards, and documentation.

Making the Right Decision

If the decision has been made to delegate the training to someone outside the organization, how do you find a good trainer? The best source for this information remains word of mouth. Someone who has done a good job for someone you know is your best bet. Always check references.

Here is a basic checklist of questions when contacting a trainer's references:

- Was the trainer well prepared?
- Were there good handouts?
- Did the trainees come away with the expected proficiency?
- Did the trainees like the instructor?
- Would you hire the trainer again?
- Would a trainee go to another course offered by that person?

When screening trainers, ask these questions:

- Does the trainer have a standard course?
- How well does it fit with your established goals?
- How many times have they taught the course?
- Are there prerequisites for a given course?
- What kind of manuals and or handouts are used in the course?
- Are there hands-on assignments?
- Is there a particular emphasis that distinguishes their training?

And, if classes will be held at your site, find out if the instructor requires a classroom with terminals and what other equipment is needed, such as an overhead projector or printer.

Continued on page 38

A trainer really should be flexible enough to present material in a variety of ways. People learn differently; an approach or example that works well for one person may not work at all for another. A visual learner (someone who learns best by seeing concepts) will require a different approach from an aural learner (one who learns best through hearing presentations). This is one of the reasons I recommend course work, and good manuals and documentation. The combination should cover the needs of both groups of learners.

Computer-based instruction (CBI) is another possibility for presenting information on the M language. There are courses available through a few sources, among them professional associations and software vendors. CBI takes the learner through material on a computer, describing concepts, giving exercises and tests, and recommending review or progress based upon the results of those exercises. CBI offers plenty of help and prompts along the way, but it works better for some people than others. Aural learners, in particular, may not find it rewarding. It is important to have an experienced, enthusiastic M resource as a backup for someone using a CBI tool; if something was unclear from the presentation, review may or may not clear up the problem, and access to someone who can explain and/or demonstrate using another approach may be more successful.

Workspace and Code Review

Whichever method of training you select, make sure that your trainees have access to good examples of M code, and that they can work in an area that will not affect your users. It is best to create a workspace (a training directory) with a subset of routines and globals separate from your system. Clean it out and refresh the data and routines regularly. Otherwise, you will have newcomers emulating newcomers' code. Newcomers must not work with the code that you distribute to your customers nor with the code being developed by experienced programmers. You run the risk of distributing damaged code or of losing the work of a senior employee. Simply put, the production and development directories are dangerous playgrounds.

Code review is an important part of training for everyone, newcomers and experienced people alike. Yes, that sentence probably of alarmed half the readers. That's good. If your code cannot stand scrutiny, it probably cannot be maintained either. Code review, however, is particularly important for newcomers. Because coding habits are formed early, you want to catch misconceptions and inefficiencies before they become a problem in your computer system. It is not enough to test to verify that the code works. I have found working code that exercised the system five or ten times harder than necessary, and even code that worked *only* because of a bug in an M implementation.

Trainee Selection

Once you've set up a training program and a place for trainees to work, whom do you train? If you can, start with your existing staff. Take the time to enlist staff before bringing newcomers on board. Without staff support, the program is doomed. Programming staff may not need the introduction to M programming, but should understand the goals of the training system. Your people should use any new procedures or standards that have been developed to support the program. If they have problems with parts of the program, take time for review. The opinion of colleagues is important to new staff. If the message is that the program is flawed or useless, putting together the program will have been a wasted effort.

Now comes the fun part. Whom do you select to train as a new M programmer? I am certain that the editor will receive a number of letters on the opinions expressed here. There are exceptions to every generalization. Having written that, here are my general rules: Good prospects include recent college graduates who know Basic or Pascal, and expert users of your system who have taken system-design or programming courses. Also included in the promising category are people who have been working in detail-oriented jobs who want to program; musicians, library catalogers, and bookkeepers are good examples.

Why are these people good prospects? People who know Basic or Pascal are more likely to be oriented toward producing applications. Pascal requires a structured approach to coding, which usually produces more legible code. Expert users are those who use an application and know all of the tricks. They know the operation from the application side, and are good resources for design review and questions about why the user wants a particular functionality. Finally, people who have worked in detail-oriented jobs understand that a character is important; they will have the patience to create code and to debug.

The bias toward college graduates is quite personal; I expect professional people to be able to recognize basic concepts in philosophy, mathematics, and literature. I have a marked preference for people who can write well in English. Tomorrow's junior programmer probably hopes to become a systems analyst in a few years.

Also possible, but more problematic, are people with lots of background in computing—computer science graduates, ATTENTION M(MUMPS) PROFESSIONALS

CURRENT M(MUMPS) EMPLOYMENT OPPORTUNITIES

Permanent and Contract, Throughout the U.S.

** partial listing **

Henry Elliott and Company Inc., professional recruiters specializing in M technology permanent and temporary computer information systems placement, is currently looking for M(MUMPS), MIIS, and/or MAGIC Programmer/Analysts and Project Leaders for its client companies.

M(MUMPS) permanent positions:

- Programmer, Programmer/Analyst, Senior Programmer/Analyst to 52K, Project Leader to 55K - Multiple positions exist in the greater D.C., Balt., MD, VA areas.
- Programmer/Analyst to 45K (PA)
- Programmer/Analyst to 40K (TN)
- Senior/Analyst Programmers to 45K (north of Boston)
- Programmer/Analysts 2 plus years to 42K (Boston)
- Programmer/Analyst to 48K (Cambridge, MA)
- Programmer/Analyst 2 plus years to 45K (NH)
- Programmer/Analyst to 37K (VT)
- Programmer/Analyst to 45K (Chicago, IL)
- Programmer/Analyst to 45K (Dallas, TX area)
- Senior Programmer/Analyst to 48K (San Francisco/San Jose, CA area)

Consulting/Contract Assignments:

- PA, 3 months + or - $28 to $40 per hour.
- New York City to $40 per hour.
- TX, long term assignments to $32 per hour.

For More Information Please Contact:

HENRY ELLIOTT & COMPANY

70 Walnut Street
Wellesley, MA 02181
617 239-8180
FAX 617 239-8210

* Partial Listing. All Fees Are Paid By Our Client Companies.

4GL programmers, and FORTRAN programmers among them. These folks usually want to write compilers, device drivers, and tools. The simplicity and elegance of the M language does not tend to appeal to them. They want more "challenge." An X-window fanatic won't be happy with the simplicity of the M windowing application programming interface (MWAPI) either.

At the bottom of my list of promising M programmers are people who thrive on complexity in computing: assembler language programmers, C language programmers, and UNIX wizards. Also included in this category, but for a different reason, are programmers with many years of COBOL experience. I know one COBOL convert to M who understands the utility of indirection. Most of the others will write ten pages of IF statements instead. In fact, if you're going to hire people with extensive experience in another programming language, be ready to pay extra attention in their transition from wizard status back to novice. Their vocabulary may not match yours, their expectations of the language will be different, and their frustration level is likely to be very high.

Again, the categories above are very subjective. Look for someone who wants to do the work you have to offer, and is willing to learn the tools of the trade and use them. While you're at it, ascertain that the person wants to work with you, and is willing to follow your rules.

Once your training program is established, how do you keep on top of developments in the language? You're off to a good start; you're reading *M Computing*. Watch for articles on the work of the MDC, and articles by expert users. Route your copy of *M Computing* to your staff with notes on recommended articles. Attend professional society meetings at the local and national level. Take along as many of your staff as you can afford to support. The information available at these meetings ranges from vendor demos to presentations of the latest projects undertaken by the MUMPS Development Committee (MDC).

The most important part of training is recognizing that everyone needs it. The second most important part is getting started.

The author is the founder of C Schell Systems, an M consulting firm in the Boston area. In the past twelve years, she has produced code to run libraries, law firms, development offices, missiles, and medical departments. She also chairs a subcommittee of the MUMPS Development Committee. Her e-mail address is cschell@world.std.com.