

# Where Do We Go From Here?

*by Thomas C. Salander*

In *The Hitchhiker's Guide to the Galaxy*, The Book (one of the characters) describes the galactic emperor as "dying, and he had been for centuries." Of course, how the statement was phrased betrays The Book's editors' prejudice. From another point of view, the emperor was extremely long lived.

M is long lived, as programming languages go. It has been around since the late sixties and has grown from a laboratory research project to a billion dollar a year industry in 25 years. Most of you reading this make your living from M. Some of you have spent your entire professional career here. Some of us know that M is on the verge of a major boom. We have seen the big break for M "just around the corner." And we've been seeing it for years.

I have a friend who is fond of reminding me that we all have a terminal illness: life. Wade is not particularly cynical. He does like to poke at my prejudices and challenge the assumptions I make when I take some point of view. He knows that any point of view has unconscious assumptions and that my unconscious assumptions will limit my options. None of us who have been the focus of Wade's challenges have always welcomed his efforts. Some of us have built such strong defenses around our parochial view of the world that even Wade's subtle pokes and prods, or overt slashes and jabs, only harden our resolve to not change. Occasionally, some of us get a glimpse of something different.

The emperor is dying, the emperor is extremely long lived. M is an archaic technology, M is on the verge of a major boom. Are you a pessimist or an optimist? Is there another option?

Let's poke a little.

## Where Are We?

M is not mainstream computing. It is only taught in a handful of schools. No major computer training firm has any offering on M. No major publication routinely mentions M, let alone uses M in its programming examples. The national meeting of the MTA draws fewer attendees than the northeast regional meeting of the SAS (a proprietary data analysis package) Users' Group. The number of vendors of M is shrinking as are the advertising budgets for those that remain.

Or . . . after ten years of high-quality, cost-effective computing in over 170 Veterans Administration hospitals, the Department of Defense has begun installing M as the primary computer language within its hospitals. Software houses like IDX and Sentient continue to expand and thrive using M as their primary development tool. In Europe, where M has shed the "medical language" image, M successfully competes with the other programming environments with predictions of much greater growth in new installations over the next five years. China has requested assistance in evaluating M as a principal computer language for the country, a potential market larger than the current worldwide installed base.

## How Did We Get Here?

M did not come out of theoretical academic development (like PASCAL). Nor did it arise as a way to convince managers that they could still oversee their programmers' work (like COBOL). M was developed partly as a "what if" and partly as a pragmatic approach to solving a relatively new problem for computers: managing large amounts of string data. A side effect was a computer language that allowed, even encouraged, rapid development of applications without dictating style or structure. It was free-form. It was radical. It was anarchy.

The academic community never embraced M, finding the whole premise alien. Scholars were more concerned with stylistic correctness and computational purity than with getting a job done. Like the old joke during the height of Artificial Intelligence's fashion popularity—if it works, it's not AI—M suffered because it worked.[1]

M also suffered because its developers ignored the "correct" classification boundaries. All software was either an operating system, a programming language, or a database manager. Except M. Initial implementations were designed to be all three. Like a platypus, M did not fit into the artificial classifications created by the academics and accepted by the industry. Without a convenient pigeonhole, M was often ignored.

It found favor with the uninitiated—the people who came to computers from other fields and viewed a computer as a tool for accomplishing work, not as an end unto itself. Without the strong, preconceived notions of what made a "correct" programming environment, they were able to appreciate M's unique capabilities and high productivity.

With their highly trained minds, the academics could not see the value of M. It was left to the musicians, lab techs, secretaries, and English majors to discover the ignorance principle: "I didn't know it was impossible when I did it."

The big computer vendors sold software as a way to encourage hardware sales. Again, M did not fit into any predefined niche. The closest any big vendor came to embracing M was Digital Equipment Corp. Unfortunately, M spent most of its time languishing in the medical systems section rather than being marketed as a data management product. From time to time people have speculated about this situation. The most commonly proposed explanation was that Digital is principally a hardware vendor. An M solution for a customer would usually be significantly less expensive than some other system. Different speculators have proposed different details. In some cases just lower total sales were enough to discourage possessing M. In others, credit was given only for *hardware* sales, but the M solution would usually require significantly less processing power so the end result was the same.

Of course, there is also the explanation that the majority of Digital's management, like most of the computer industry, categorized (and dismissed) M as a "medical language."<sup>[2]</sup> However, the low cost of M solutions was a real problem when trying to market M to established IS managers. Like big cars, big guns, and big muscles, having a big computer was, sometimes, viewed as a status symbol (at least to the person who had it). The idea "smaller is better" (in computers) is a recent phenomenon. In the early seventies many large businesses and institutions would have a whole floor just for their computer. Some managers would boast that they had to have a new building added, the size of a small house, *just to hold the cooling equipment*.

When prestige is important (and it is to many people), replacing that huge mainframe with a relatively small set of mini-computers is not appealing. Add to that the reduced budget for maintenance and power consumption, and soon buying M is a visceral threat, not a logical choice.

There is another factor that may have led to the typecasting of M. The medical community, particularly hospitals, was not as quick to automate as were other industries. While there might have been a computer in the accounting department to run the payroll, the actual business end was still a manual process. While a few small segments of the medical community may have been doing well, for the most part money was tight. If automation was to come, it had to be cheap.

Thus, for the same reason M might be rejected in a corporate IS center, it would be welcomed in an overworked medical

records office or fledgling laboratory automation effort. With fewer egos (and jobs) to threaten, fewer preconceived notions to overcome, and an origin designed to meet their specific needs, M found greater acceptance in the medical community.

## Now What?

A long time ago I lost a canoe race. The race was for charity. Neither Brian nor I were canoers, but we knew enough to get by. We borrowed a canoe and paddled onto the Susquehanna for the first time just before the race began.

For two guys who hadn't been in a canoe in years, we were doing OK, settling into the middle of the gap between the lead pack and the middle pack. After a while the river forked around an island. The leaders were going to the left. On the bridge before the fork stood my wife, pointing to the left. We went right . . .

. . . and ran out of water.<sup>[3]</sup> I tell this story for a couple of reasons. First, I am not a soothsayer. If I could accurately predict the consequences of any given action, my life, and the lives of several other people, would be a whole lot different. Second, an informed guess is better than an uninformed guess, but it is still a guess.


Well then, what are our options? If M is on the decline and survives primarily on the momentum of its installed base, then we can look forward to a steady decline in opportunities. Vendors will look to increasing their share of a shrinking market, like buggy whip manufacturers in the age of the Model A. Programmers will move around in a real game of musical chairs, with fewer chairs each time we look around. Managers will either feel relieved that they no longer have to push to stay ahead (if they figure they'll retire before they finally have to port to something else) or will be nervous because they fear bringing in technology for which they will not be the source of all expertise.

This option is really the easiest to accept. It requires no effort to make it happen and it allows us to continue doing just what we've always done. We don't have to change, though change may happen to us, like the gazelle that ventures too close to the edge of the herd when the lioness is hunting.

Another option is to focus our efforts on the medical market and stop trying to be a general data management system. M's origins are medical. A large portion of its installed base (in the USA) is medical. We've had good success in penetrating the medical market.

Unfortunately, some of the reasons M was initially successful are no longer applicable. Where once we were entering into an area with little or no computerization, most medical envi-

## What is a Gadget? What kind of Values can I assign?

Answers to these questions and more are just a few  clicks away.

New CBI now available . . .

### MWAPI Reference & Tutorial CBI

An On-Line Interactive Tool for learning the M Windowing API! The **Tutorial** guides the programmer through a Comprehensive Tutorial that covers all important features of the MWAPI. Exercises are presented for all gadgets. A completely functional windows application is built. The **Reference** contains the full MWAPI specification in hypertext format. It can reside in a Window for quick visual access to gadgets at all times.

ESI also has other CBI's and Lecture/Workshops available in M Programming, File Manager, MSM, DSM, DTM, Object-Oriented Programming, VMS Concepts and EsiAuthor.

**Call ESI for more details!**



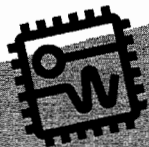
**EDUCATIONAL SYSTEMS, INC.**  
5 Commonwealth Road • Natick, MA 01760  
Tel: (508) 651-1400 • Fax: (508) 651-0708

## The answer to all your M computing needs

- ✓ M (MUMPS) and C Programming
- ✓ New Application Development
- ✓ Software Maintenance and Enhancement
- ✓ M Programming Classes
- ✓ Management Reporting
- ✓ Software Validation
- ✓ Experts in Hospital Information Systems

We are knowledgeable in FDA regulations and federal guidelines concerning good manufacturing practices. With over 15 years experience in the M environment, we offer solutions at competitive rates.

**out west**  
CONSULTING, INC.



2450 East Speedway Blvd. Suite 5  
Tucson, AZ 85719  
Tel: 602.795.7076 or 800.2outwest  
Fax: 602.322.0781

ronments today have access to a variety of computers, computer languages, and computer database systems. Where the MIS department used to be a secretary, a lab tech, and an intern, we now deal with established professionals who often have come from nonmedical computer environments. The myth that "medical computing" is somehow different from other computing is being laid to rest.

In addition, with a more established MIS staff, the herd instinct in the medical MIS departments is growing just as strong as it is in MIS departments generally. People want to be different, but not *too* different. The further we get from the norm, the more nervous we tend to get.[4] When people look around to see what other MIS departments are doing, they see Oracle and Sybase and systems written in C++ and Visual Basic. If we try to sell them an M solution they may say "what's that?" (with the apprehension reserved for door-to-door salesmen). When we tell them M has been around for 25 years, their next question is likely to be "then why haven't I heard of it?"

In the end, focusing on the medical market is really not specializing, it is putting blinders on. It also sounds suspiciously like sour grapes.

## The Hard Way

When Wade pokes at me, he isn't telling me what decision to make. What he does is encourage me to be honest about what I've decided, to myself if no one else. If I choose not to do something, I may say "I can't" when more likely the reason is "I'm afraid I'll fail" or "I'm afraid to change." Knowing it is fear that stops me, not capability, helps avoid the layer after layer of deception I'll have to build to support my excuse if I don't proceed. It also puts me in a better position if I do.

If M is to survive and flourish, we cannot continue to make excuses. It will also take courage, hard work, and large amounts of brutal honesty.

Some thoughts on the details:

**Stop thinking we're superior.** As a community we have an arrogant streak as if we are better for using a superior technology. We're not and it's not. M has many good points: high productivity, lower hardware requirements, good scalability. But M also has some weaknesses: low transaction reliability, character-based screens, poor integration with other environments, and few development tools (most of these have been addressed in the new standard, but the installed base does not reflect this). Like a parent, we tend to highlight M's strong points and minimize its weak points.

The fact is that M is not superior to another environment like Visual Basic, nor is Visual Basic superior to M. They do have differences, each with its strong and weak areas. If we continue to ignore that fact, we will not put the needed effort into improving our weaknesses.

**Stop the hype.** M is not a religion, it is just one of many options available for developing computer applications. When we proselytize M to the uninformed as if we are trying to save their souls, we will usually only succeed in convincing them we are a bunch of nuts.

We also tend to overstate our case. *Understatement* will do us better in the long run. If we tell new customers that they can run fifty users on the system, and they find they can run sixty comfortably, M's image will improve in their minds. If they find out they can only get fifty users on the system if a snail's pace is acceptable performance, we confirm their basic suspicion of our other glowing reports.

**Stop marketing to shoestring budgets.** One of the original advantages of M was its low cost. Unfortunately, we now find it difficult to bring in the revenues we need to continue development of the systems and provide the needed customer support. Those of us who purchase systems need to stop expecting to run a multimillion dollar company on an M license costing \$500. There is still some truth to the adage "we get what we pay for." Go for quality. M still has many advantages and still is a low-cost solution. We can accept a higher cost and still be low cost, particularly if the quality improves.

Implementors need to stop assuming the price is the primary factor in purchasing a system. Quality is important. Quality in the product, not the sales hype. People will pay more for a better product, particularly if they are still paying less than they would for a competing product.

**Quality is important.** Yes, I said that already, but it also needs a heading all its own. Maybe we've forgotten the point. Quality is an issue of every aspect of M Technology.

Implementation releases need real quality control, not just beta testing. Tech support needs the same status (staff load, quality of training, level of experience) as development. Application developers need analysis and design, not hack and slash. They also need to recognize programming as engineering, not craft.

**Think as a community, not as competitors.** More effort goes into stealing each other's customers than into creating more customers. What good will it do any vendor to be the largest if the customer base is shrinking? The current work with COMDEX is a good start, but it is only a start. More cooperation between implementors must take place.

But it is not just the implementors. The software developers need to look beyond the M community. If you write a tool, are you writing it to compete with (and take customers from) another M developer, or are you writing something that would also be useful in the non-M market?

End users also need to be community-minded. We need to be involved, through MTA, MDC, LUGs, writing articles, and any other way we can interact. The more we are involved, the more other nonparticipants will start to become interested and add their efforts. It may be that only one in ten M users are currently involved in any way. If we even doubled that number, we would be well along as a strong community with something to offer the rest of MIS.

**Watch the competition.** Not us, them. Windows, PCs, networks, interoperability, reliability, customer support—these are just some of the focus points in MIS, outside the M community. There is where the competition lies. While we may have a superior technology for data handling, the customers may not know that, and may count it as just one more piece when making their final decisions.

**Developers need executables.** With most other environments there is the development system, and then there is the product you produce from that system. Your end product is usually an "executable" and, usually, it can be distributed freely.

If we want to get M out there as a real alternative for software development, we cannot continue to shackle the developers by insisting that they distribute full licenses. The end user does not need a development system (which most M systems are), and the cost makes M the *high-priced* option, not the low cost we have come to expect.

**Developers need tools.** The problem may be chicken and egg. Developers do not use many tools partly because there are not many tools to use. There are not many tools because there hasn't been much incentive to market them (few buyers). Since we have to start somewhere, I go for the buyers. That is, the software developers need to start using the tools that are there. As they get used to the idea there will be more market for better tools.

Of course, getting programmers to use someone else's tools leads to the next point:

**Purge the NIH (Not Invented Here) syndrome.** If every application programmer is going to insist writing everything from scratch, M will be the computer equivalent of quilting: it's nice busy work that keeps a person occupied (presumably

pleasantly), its end product may be pretty and individual, but if you need to keep warm, the more off-the-shelf components the better.

**Keep trying.** Few things worth while come easily. When each of us started in this world we had a built-in tenacity that served us well. Anyone who has watched a child learning to walk can attest to that tenacity. Few of us are willing to fail as many times as that child does and still keep trying.

But we need to try more. One failure, two failures, these are not the things that make or break. Ten failures, twenty failures—salesmen have told me that a success rate of *one in twenty* is doing well. Probability theory shows that all the prior instances have no bearing at all on the probability of the next try.

M has been around for a long time. M is a new technology. It depends on your point of view. It's all old, has been tried before, and we know it won't work. Everything is new and fresh each time we try.

Which point of view you take decides what happens next. ■

---

Thomas Salander is the Vice Chair of MTA, and headed the MUMPS Development Committee from 1988 until early 1994. He has been writing for *M Computing* and its forerunners for several years, having served as guest editor and on the editorial board. He can be reached at Thomas Consulting, 3629 Kimble Road, Baltimore, MD 21218-2027.

---

## Endnotes

1. In 1969 I was taking a programming course in Fortran on an IBM 360. This was a religious experience. The keypunch machines were the holy water with which we anointed ourselves on entering the great hall. The computer operators were the acolytes who took our offerings to the altar. The systems programmers were the high priests who were the few who could interpret the gods' utterances, who informed us (through the acolytes) that all the failures were due to our own imperfection, and who could bless you with a dispensation (and run your program with the right switches set) if you knew them, they liked you, and you delivered a six-pack by Friday night. The Information Systems faculty were the philosophers, sages, pundits, and gurus.

Once we were handed a deck of punch cards and told to sort it. After a couple of minutes I handed the deck back, sorted by hand. The instructor was not amused. Three days later most of us had succeeded in getting a program to run. When graded, there was no correlation between a program's ability to sort the deck and the grade received. Like apprentices to a master artist, emulating the instructor's *style* was a better road to success.

Even then, an engineering approach to programming was more lip service than reality.

2. I find this explanation the most plausible, primarily because of Occam's Razor.[5] However, some may argue that this explanation does *not* cover all the known facts (as required by Occam)—in particular, the evidence that unit performance appears to have been hardware-sales related. To these people I would add the corollary to Occam's Razor: Never attribute to malice that which could also be explained by ignorance.

3. Actually, the worst part was the mud. When we got out to try and get back to usable water, we went up to our knees in river mud. Despite this, we still finished the race near the back of the middle pack. We learned much from this. Brian learned to wear a hat on his bald head when he was out on the river. I learned not to be so quick to jump out of the boat. We also learned that taking a different path doesn't always put you ahead, but often gives a more memorable experience.

One more odd bit: we were the second canoe to take that fork. Later we learned that nearly a third of the middle pack followed *us*. This included at least one team that regularly canoed this river. Interesting thought.

4. The problem is related to perception. (This is part of what advertising is designed to do: convince you that "everyone else is doing it.") We perceive a higher intelligence level in the mass of people than in ourselves. I started using WordPerfect in 1988 as my first PC-based word processor. While I still use it from time to time, I am not fond of it and have found several other word processors that I like better. Yet people continue to buy WordPerfect primarily for one of two reasons: 1) they know how to use it; 2) it is the largest-selling word processor, so it must be good.

Thus, one of the reasons WordPerfect is the largest-selling word processor is because it is the largest-selling word processor. (Momentum in motion.)

The same process is happening with programming languages, where "object-oriented assembler code" is becoming the odds-on favorite for the next programming language of choice. This would replace COBOL as the most required language skill. However, COBOL *still* dominates, primarily because of the huge installed base. (Momentum at rest.)

5. William of Occam (c. 14th century), "Entia non sunt multiplicanda praeter necessitatem."

Translation: "Entities ought not to be multiplied except from necessity."

Explanation of the translation: "All unnecessary facts or constituents in the subject are to be eliminated."

Rephrasing of the explanation: "The simplest explanation, provided it accounts for all the pertinent facts, is usually the best."

Common (but incorrect) usage of the rephrasing: "Don't make it more complicated than it has to be."

## Moving?

Let us know! Contact us at:  
M Technology Association  
1738 Elton Road, Suite 205  
Silver Spring, MD 20903  
Phone: 301-431-4070  
Fax: 301-431-0017

Please include your MTA ID number or mailing label from *M Computing* along with your new address.