Dollars and Sense: How M Compares with COBOL and Relational Database Software

by Don Gall

Abstract

How do you compare computer languages? To quote from Browning, "Let me count the ways." We have all seen comparisons based upon the number of lines of code, the length of time to generate an application, the sort time, the execution time, and a wide variety of other quantitative benchmarks.

Is the choice of a language a factor in programmer productivity? The cost in programming and programmer support time certainly plays a major role in the cost of an application. Yourdon states that the programming language is not a major factor in programmer productivity, while Martin makes a case for the language having a large impact on productivity. [1,2]

This article attempts to compare M with COBOL and a variety of implementations of relational databases from an economic point of view by asking how much it costs to generate, support, and implement an application.

Introduction

Selling M Technology against other languages has always been an interesting challenge. Eighteen years ago, industry consultants could not understand why a language other than COBOL or RPG would be used to write a commercial accounting application. Even today, there are those who believe that if it is an accounting application, it should be written in COBOL and run on an IBM AS/400. As relational databases came into vogue, it became necessary to defend the use of M Technology on a second front. The new group of consultants could not understand why an old language based on an outmoded, hierarchical data structure would be chosen over the obvious wave of the future, a relational database.

The purpose of this article is to present the arguments and evidence that lead us to the conclusion that M continues as the language of choice for this general class of software. One way of comparing the relative merits of programming languages is the employment of benchmarks. Theoretically, carefully designed benchmarks should provide the best measure of comparison between two languages. Unfortunately, very carefully designed benchmarks can also prove almost any point that you want to make. This article compares the end results of the efforts of competitive software companies that have elected to employ different languages. It is not surprising that the comparison of these end results can be put into one word, money. In simple terms, a software company must deal with the cost of development and support, the cost of implementation and the cost of sales. Each of these costs is affected by the choice of the computer language.

A goal of this article is to quantify these cost differences between M and COBOL and between M and implementations of relational methodologies. To do this, in the absence of detailed financial statements, it has been necessary to relate costs to the size of the programming staff or the total number of employees. In many other industries, this assumption would lead to erroneous conclusions. The single biggest cost to a software company is payroll and a good measure of the success of a software company is the ratio of total sales to the total number of employees. Very successful companies have gross sales of $150,000 to $275,000 per employee. A company with sales much below $100,000 per employee is generally not making a profit.

In the following comparisons, the twelve major software companies which compete for business in mid-size law firms (20 to 200 attorneys) are examined. One of the companies (Omega) uses M as a language, three use COBOL, two are in the process of converting from COBOL to a relational database implementation and thus have a mixed customer base, and the remaining six use various languages to implement their relational databases. We recognize that the results reported here, based upon only twelve software companies, can in no way be regarded as statistically significant. The trends and the consistency of the data are of interest, however.

Historical Background

Omega Computer Systems, Inc., is an M Technology software company. Although Omega's first nonmedical application of M (Meditech MIIS at the time) was for a law firm, the first ten years of Omega's history were spent developing custom software for a wide variety of medical and business accounting applications. In the past eight years, Omega has

specialized in accounting software for law firms. The major components of accounting software for law firms—accounts receivable, accounts payable, and general ledger—are similar to those components of many other businesses. Although the observations and conclusions expressed here are the result of experience with law firms, the general nature of accounting business applications should make these observations and conclusions applicable to a number of other businesses.

At the time of Omega's first law firm installation, eighteen years ago, all of the other companies specializing in law-firm accounting software in the Western United States programmed in COBOL or RPG. These two languages continued their dominance in the law firm industry until software employing relational database methodology began to appear about ten years ago. Today, only three of the twelve major software companies in this sector continue to use COBOL. Let us first examine some of the reasons for COBOL's decline in this market.

M versus COBOL

A number of authors have reported that COBOL requires a significantly larger amount of programming resources for an application than do many other languages. [3,4] In fact, James Martin's definition of a fourth-generation programming language is: "A language should not be called fourth-generation unless its users obtain results in one-tenth of the time with COBOL, or less." [5]

The increase in programmer time is also supported by the relative sizes of the total software packages. While Omega's total M source code takes up less than 2 megabytes, the programs from several competitors using COBOL occupy a minimum of 60 megabytes of disk. If the time required to program an application were proportional to the size of the resulting programs, this one example would place the COBOL to M programming time ratio at 30. If an old IBM rule of thumb that the time to write a program is proportional to the square of the number of lines of code is accurate, the ratio of 30 would be conservative.

A significant economic measure of programming efficiency is the relative sizes of the programming staff of law-firm software vendors using COBOL and M. Omega has a programming staff of five development and support programmers with over 135 law-firm clients or 27 clients per programmer. At its peak, the most successful COBOL law-firm software company had 100 programmers supporting 350 clients. This ratio of 3.5 clients per programmer was made worse because all but a few of these 100 programmers were in a support role. The company's inability to produce new versions and its high

costs eventually led to its demise. This company and one other COBOL firm are no longer active in this market and are not among the three COBOL firms cited.

Since it is usually easier to get a count of the total number of employees of competitors than it is to determine the size of the programming staff, let us look at another economic measure, the ratio of clients to the total number of employees. The American Bar Association publishes an annual directory of companies marketing computer software and services to law firms. [6] All information in the directory is supplied by the vendors. One of the questions asked of vendors is the number of installs. It is obvious in some cases that certain vendors interpreted this to mean the total number of installations of their software, while others, including Omega, listed the number of active clients. In all cases cited, we have used the term *installs* to mean the number of active clients, which in some (non-M) cases will produce a client to employee ratio higher than it really is.

A second possible source of error is in the size distribution of clients. Larger law firms require more support and modifications than do smaller firms. Since the *Locate* directory of the American Bar Association does not differentiate, I used only those vendors specializing in mid-size firms (20 to 200 attorneys). Note that using this information, however, does not completely remove the bias introduced by the variations in the distribution of law-firm sizes among the twelve software companies.

While Omega has consistently operated at about 10 to 12 clients per employee, the *Locate* directory indicates that the COBOL software companies range from 1.9 to 3.6 clients per employee. [7] (The former competitor cited previously had 225 employees and 350 clients, a ratio of about 1.6.)

These rather consistent client-to-employee or client-to-programmer ratio differences indicate a dramatic decrease in the development and support costs for M Technology as compared with COBOL. This is no surprise to the M community or those who have already moved away from COBOL.

The cost of implementation also has been a factor in the movement away from COBOL in the legal community. Omega's experience with law firms indicates that for every dollar a law firm spends for accounting hardware and software, it spends $6 to $10 for word processing. Even as recently as five years ago, the dominant players in law-firm word processing were companies with proprietary software running on proprietary hardware. Today, it seems there are only two types of law firms, those that have a local area network (LAN) and those that soon will. This rapid movement to personal computers (PCs) and networks has been a definite advantage for Omega because of M Technology's proven ability to operate on Intel 80486 chip servers in a client/server structure on networks. The hardware traditionally used in the COBOL environment is neither as readily adaptable to LANs nor as inexpensive as the Intel 80486-based PCs. In one specific instance, the choice came down to spending more than $100,000 for hardware for a COBOL solution or buying a 486 PC for an M Technology solution. The software prices were essentially the same. It is of interest to note that the consultant recommended the COBOL solution. Instead, the client chose to save more than $90,000.

Measuring the cost of sales is more than adding up the salaries and expenses of the sales staff. It must somehow also include the time, effort, and expense of losing sales a company should have won. Five or more years ago, Omega experienced a number of lost sales because COBOL was mainstream technology and M was not. COBOL and accounting were synonymous to most consultants. That M was faster and somewhat less expensive did not offset the comfort many had with COBOL. The movement to PCs and LANs has made M Technology much faster, much better, and a whole lot less expensive.

The decline in the use of COBOL for law-firm accounting also is indicated by the year-to-year changes reflected in the *Locate* directory. In successive years, one of the three long-time COBOL software companies went from 306 employees to 216 and another went from 56 employees to 31.

M versus Relational Database Implementations

Using Martin's definition of a fourth-generation language, M and most relational languages certainly qualify as 4GLs. It is not obvious that a good case can be made for expecting a significant increase in programmer productivity for M over a good relational database language. On the other hand, the *Locate* directory indicates that the ratio of the number of clients to the number of employees for the eight companies now using relational methods ranges from 1.8 to 4.4. The two companies that are in the process of converting from COBOL have ratios of 2.4 and 2.9. One company, which has written its own relational database system in Pascal, is at 1.8. The remaining five companies, which have been using a relational database for longer, have 3 to 4.4 clients per employee. This is still considerably fewer than the 10 clients per employee of Omega.

Program size and program disk storage requirements appear to vary widely among relational database products. The system written in Pascal has more than 100 megabytes of programs while others are closer to Omega's 2-megabyte value. Thus, differences range from very significant to insignificant.

The hardware requirements also show a wide variety among the different relational languages. While Omega has operated effectively on 386 and 486 technology (on computers costing less than $10,000), competitors have quoted computers ranging in price from $25,000 to $175,000. The two consistent differences are the disk storage requirements ranging from two to four times that required by M Technology and the superior performance of M Technology systems. The key factor in these two differences is the fundamental difference between the hierarchical structure used by M and the table structure of a relational database language.

The major advantage that relational databases have over hierarchical databases is that all information is stored in tables and that all tables have a single general structure. This enables the developer of a relational database language to create general programs or commands to create, edit, delete, search, sort, and print reports for the one general structure.

The major advantage that hierarchical databases have over relational databases is that information can be stored in many different types of structures. This can make life more difficult for the applications developer and it is very likely that this is why very few hierarchical languages have had a major impact upon the database market.

Relational databases employ a table or flat file structure consisting of columns and rows of data. Each column contains a specific data field such as a vendor identification, city, ZIP code, etc. Each row of the table is a specific instance of a set of data fields, a specific vendor in this example. There are many database examples that readily fit into this structure. Cross-reference and other key tables are used to link one such table to another and to enable more rapid searching and sorting for the tables.

What about data structures that do not fit this table structure? There are many examples, but one of the most common is the many-to-one structure common in many database applications. A cash disbursements journal is one example. A given cash disbursement has one date, one check number, one payee, and one amount but may have one or many general ledger accounts and amounts to be debited. The designer of the relational database must either create one table with a row for each debit entry, thus having one or more rows per check or must split the journal into two tables, which is the more common method. The first approach produces a large table with many redundant entries. The second approach leads to poor clustering of related data. The hierarchical structure allows a file structure of:

```
^DISB(check date,item) = payee, check number, amount...
^DISB(check date,item,debit item) = debit G/L, amount...
```

This leads to the issue of data clustering. It is obviously important to store logically related data that will often be used together as physically close together as possible. C.J. Date states, "Physical data clustering is an extremely important factor in performance." [8] If one uses an intelligent node-naming convention, a hierarchical structure can ensure that these related sets of data fields are usually adjacent to one another in the same data block. In the worst case, they will be contained in the next logical data block which is pointed to by the original block.

The disk file structure of M is capable of producing optimal data clustering. A relational-database program must do a logical join of the two tables to produce a cash disbursements report. By using the global structure of `^DISB`, an M program can produce the report from data contained in one or a few logically contiguous disk blocks with no sorting required. As a specific example of this, let us look at the cash disbursements global of one of Omega's larger clients. The client currently keeps five years of cash disbursements online. The checks are for 1,373 accounting dates and occupy 4,479 (4k) disk blocks. There are four levels of index blocks in the b-tree. In the worst possible case in which a day's cash disbursements are spread over five data blocks (the average is 3.26 blocks per day), only nine disk accesses are needed to import the entire day's cash disbursements. The disk-buffering technology of M ensures that we will need to import each disk block once. At least two of Omega's competitors would have to sort through the entire 17+ megabytes of data to find one day's cash disbursements. The M Technology solution does less disk-accessing in printing each day's cash disbursements for an entire year than the relational methods do to print a single day's records.

The variations in structure that are available in a hierarchical language have enabled M Technology to create very efficient and high-performance software. Omega has structured the client-matter file so that all demographic information, work-in-process, billing history, advanced deposit, and trust history are contained within the matter global and therefore are logically contiguous on the disk. For the average matter this means that all of this information can be brought into memory with two sequential disk accesses after the starting block is located through the b-tree. The same data, using a relational model, would require thirty-eight separate but interlinked tables, requiring many more disk accesses. Although many programs only need one or a few of these tables, the combined bill printing and billing process accesses thirty of the thirty-eight tables.

Disks are slow and CPUs are fast! An average disk-access time of 10 milliseconds is equal to only 100 random disk accesses per second. The excessive disk-accessing required by large relational databases can be very detrimental to performance. The increased use of cross-reference and clustering indexes helps the performance but adds to the disk-storage requirements. This then leads to requiring bigger, faster, and more expensive hardware.

Relational database systems are currently very popular among consultants. We have had requests for proposals that specify that the system must be relational. Fortunately, a number of M vendors have SQL packages that can be added to the software to allow the presentation of a relational view to a variety of SQL-type inquiry and reporting software packages. According to the accepted definition of a relational database, which does not specify the disk-storage methodology, M is relational.

Conclusion

M has a strong economic advantage over COBOL mainly because of M's superior programmer productivity for both development and support. M also has an economic advantage over COBOL because of the cost of hardware implementation.

M's economic advantage over relational languages is due to the cost of hardware. There also appears to be some economic advantage in programmer productivity. The major advantage of M over relational languages is in performance, which, surprisingly, is still not accepted by many to represent a significant advantage in cost.

Don Gall is the president of Omega Computer Systems, Inc. He was an associate professor of mechanical engineering and biotechnology at Carnegie-Mellon University, a research associate professor of surgery and anesthesiology at the University of Pittsburgh Medical School, and a visiting professor at the IBM Research Laboratory in Switzerland prior to starting Omega. He has been involved in M programming for twenty of his thirty-eight years in computer programming.

Endnotes

1. E. Yourdon, *The Decline and Fall of the American Programmer* (Englewood Cliffs, New Jersey: Yourdon Press, PTR Prentice Hall, 1992).
2. J. Martin, *Application Development Without Programmers* (Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1982), 2.
3. T. Munnecke, "A Linguistic Comparison of MUMPS and COBOL," *AFIPS Conference Proceedings* 49 (1980): 723-729.
4. S.H. Johnston, "The Effect of Language on Software Costs," *M Computing* 1:3 (1993): 39-54.
5. Martin, p. 2.
6. Ernst & Young, *Locate 1992-1993: A Directory of Law Office Computer Software Vendors* (Chicago, Illinois: American Bar Association, 1992).
7. Ibid.
8. C.J. Date, *An Introduction to Database Systems* (Reading, Massachusetts: Addison-Wesley Press, 1986), 50.