M Makes Telephone Directories: An Application for Typesetting and Pagination

by Mark A. Wieder

This article outlines the history of an application which has survived and prospered through several iterations of the technology revolution. The author also reviews one of the application's more exceptional aspects: typesetting and page layout.

Over a period of twelve years, Polylogics Consulting, Inc., has been developing an application named *DIAD* (DIrectory ADvertising), for use by the telephone directory publishing industry. Directory publishers generate their revenues by selling advertising space. Their systems requirements revolve around managing the sales and billing of this advertising (a common type of database service), and around the typesetting and page layout of their telephone books (an unusual and technically demanding requirement).

Due to its long history, *DIAD* has had to undergo several migrations in hardware platform, operating system, typesetting platform, and M implementation. The journey has been commercially successful, in large part due to the selection of M as the primary development platform.

Position within the Industry

Telephone directories are published by two classes of companies, utilities and independents. Utility publishers are the local phone company in the area where they publish. In the United States, the utility publishers include the Regional Bell Operating Companies (RBOCs) created by the breakup of AT&T, plus GTE and many smaller regional telephone companies (TELCOs). In many other countries, the TELCO is a state-owned or state-operated entity, and the state contracts with private companies to sell and produce TELCO directories. Often these contracts go to international divisions of the American RBOCs and GTE. A key aspect of the utility publishers is that they are directly informed by business and residential users of their connection and disconnection of telephone service. Therefore, these companies, as a side effect of their local telephone service monopoly, also have a monopoly on the timely update of telephone listings, and know first of the creation of new businesses and the demise of failed businesses. The treatment of this information in a monopolistic manner by TELCOs has been challenged as a violation of antitrust law and is the subject of several lawsuits by the independent publishers, many of which lawsuits have reached the Supreme Court.

Independent publishers are private, unregulated companies whose directories compete with those published by utilities. Independents differentiate their products from those of the utilities by cost (their advertising space is typically much less expensive), by scope (covering a smaller or wider area than the utility), and by innovations such as "talking" Yellow Pages, where the consumer gets a phone number in an ad that provides information about advertisers and their products. Independents usually have to buy listings from their competing utility(ies) in order to have complete information in their books, and to learn when numbers are "new-connects" and "disconnects." Independents also do not have the luxury of billing for their advertising on businesses' telephone service bills, as do the TELCOs. They therefore have collection problems similar to those of most businesses.

Most *DIAD* users are independent publishers. There are several competitors to *DIAD* in this marketplace, usually limited to either the PC or the Macintosh platform. M Technology allows *DIAD* to run on multiple platforms, even within a single site. The utility publishers were typically early users of mainframe computers, and as a result developed their own in-house solutions twenty or more years ago. They are now looking at new solutions such as *DIAD* and its competitors. Among outside system suppliers, the utility market has been dominated by one particular company whose systems run on VAX under COBOL, and whose products are very expensive (many millions of dollars per installation) but also very comprehensive, including such capabilities as digital storage of ads.

Portability

Because M is truly portable, *DIAD* has been able to preserve both Polylogics' and the end users' investments in software development. How many programming environments would have permitted a seamless transition through the platforms shown in figure 1?

Year	Hardware Platform	Operating System
1982–85	Data General Nova	Stand-alone M
1984–93	Data General MV series	AOS/VS
1985–91	Digital PDP-11 series	Stand-alone M
1988–present	Digital VAX series	VMS
1989–present	286/16 and 386/33 systems	MS-DOS
1990–present	PC network environments	MS-DOS, NETBIOS
1992–present	PC Windows environments	MS-Windows

Figure 1. Typical platforms for *DIAD*.

Scalability, Device Independence, Distributed Processing

The smallest *DIAD* sites function with a single PC and a few users running terminals. Larger sites run up to 128 concurrent users with terminals at multiple sites across the country, or multi-server PC networks with dozens of total users.

M's support of "dumb" terminals simplifies installation and maintenance in environments not requiring networks. At more sophisticated sites, networked M environments allow both processing and data storage to be decentralized, while gaining the networks' device- and file-sharing advantages. Dumb terminals typically are used in conjunction with networks as well.

Acceptability

It has been our experience that the use of M does not necessarily enhance the acceptability of a product in the marketplace, and often generates objections that must be overcome. This negative bias seems to be based on several perceptions, mostly not valid, but difficult to rebut. It is necessary to overcome these objections on the part of business owners and managers, some of whom lack computer experience outside the venue of off-the-shelf software. Figure 2 shows some representative objections to M-based applications and some corresponding responses.

Objection	Response
M is not a "mainstream" way to go, is it? Really a way of saying, "I don't know it and I don't see much about it in print; my staff doesn't know it, so it must have significant disadvantages."	Respond with irrefutable positive attributes: scalability, portability, speed of development; point out that M isn't owned or sponsored by any hardware vendor, and that M enables the implementation of projects on much more cost-effective platforms than such vendors might prefer.
M does not easily allow import and export of data from and to off-the-shelf products, such as spreadsheets and database programs, and thus requires "expensive" programming.	To varying degrees this objection is valid for any programming language. Discuss tools that can be layered over applications, such as report writers and tools that allow SQL access to existing data. Also, point out that M code can be developed very cheaply and quickly even without such tools.
M programmers are hard to find.	Refer would-be employers to the M Technology Association Job Referral Service; offer to help train users' in-house programmers. Point out that a real problem is that the users' programmers (in any language) don't know the application, and that most newcomers won't know the industry. If in-house programmers are to be hired by the client, who will be responsible for programming done by these programmers, especially their programs' interaction with the vendor's programs?

Figure 2. Typical objections to M-based applications and corresponding responses.

Integration, Version Control

DIAD is integrated (as opposed to being modular); that is, all its programs are designed to work together in a single database environment. As such, all its programs must act consistently to maintain the logical integrity of a database. Thus, it is critical that any event changing several pieces of data must change them all such that logical integrity is maintained, whatever that happens to mean in a given case.

Ideally, these issues are controlled by the use of centralized functions, so that any given set of data nodes can only be created, changed, or deleted by a precisely defined, well-documented set of processes. Every programming "shortcut" that subverts this goal is, of course, punished mercilessly by the end users who, faster than you can say "O-rings," will embark upon any newly available path to data corruption.

A significant part of *DIAD*'s growth comes in the form of custom development for existing users. There are several possible strategies for managing multiple sites with different, possibly mutually exclusive and conflicting, sets of custom enhancement requests. One option is to allow each site to be independent and different. This requires the least effort for each project, but means multiple versions will exist for many programs, so bug fixes and enhancements must be implemented in each version of the same program.

Another is to have a core set of programs that are universal, and allow multiple versions of certain programs. This hybrid approach inherits some of the above problems.

A third option is to maintain one and only one release set of the entire application, safely distributable to all sites. Here, controlling the activation of enhancements with switches, tables, and run-time option prompts (some controlled by programmers, some by users and their managers, and some by user process control and training) is key. This approach requires more design and programming for each change, but only a single set of code needs to be debugged, tested, maintained, and documented. (Of course, there are many combinations to test.) *DIAD* is maintained via this method. All implementation-dependent code, such as code relating to device parameters, is centralized and controlled by switches.

Internationalization

On several occasions, a very large corporation in the industry has called upon Polylogics to adapt *DIAD* to create prototype pages in foreign languages, notably Hungarian and Polish, in a matter of days. (At these times, of course, any concerns about the use of a "strange" programming language have gone by the wayside.) In spite of the considerable technical resources available within such companies, *DIAD* was their quickest path to a solution. At such times the development speed and flexibility of M really come to the forefront. M's ability to support foreign character sets is critical, as is the fact that sorting names in *DIAD* is controlled by a central function (sorting of names in the above languages sometimes treats character pairs and accented characters as a different, single character for sorting purposes).

The Yellow Page Pagination Problem

Figure 3 labels the regions and objects that appear on a typical telephone book Yellow Page. The process called "paging" refers to the laying out of these objects on each page, such that a number of criteria are met. Some *DIAD* users produce more than one hundred telephone directories a year. For such companies, the creation of pages becomes a manufacturing process, with exponential economies of scale to be realized with each step of automation.

Paging can be viewed as a version of the tiling problem: how can a "floor" be tiled with a given set of rectangular tiles of a given size. There are, however, many constraints on the Yellow Pages tiling problem. Following are several examples of page problems that must be solved regularly.

Listing blocks must, of course, appear alphabetically. There must be no widow or orphan listings at the beginning or end of a column.

Display ads must be placed in order of seniority. Seniority is based on several criteria:

- Larger ads are senior to smaller ads.
- Within size, ads for advertisers appearing longer (over prior issue years) within the heading have seniority. (There are numerous variations on this theme, including ignoring it.)
- Within seniority date, alphabetic name is the tie breaker.
- Ads that are more senior do not simply appear "ahead" of those less senior. The outer channels (those further from the binding) are considered more desirable positions. Upper positions on the page are preferable to lower. Right-hand pages are preferable to left. Certain advertisers may be specifically promised certain positions (i.e., upper right-hand corner of right-hand page) for their ads. Thus, it is necessary to consider the entire spread (left and right facing pages) when making paging decisions.

In general, there may be no widow or orphan display ads (ads alone on their spread or on their page relative to other ads in their heading).

Some publishers forbid any page to have only ads; any page with ads must have listings also.

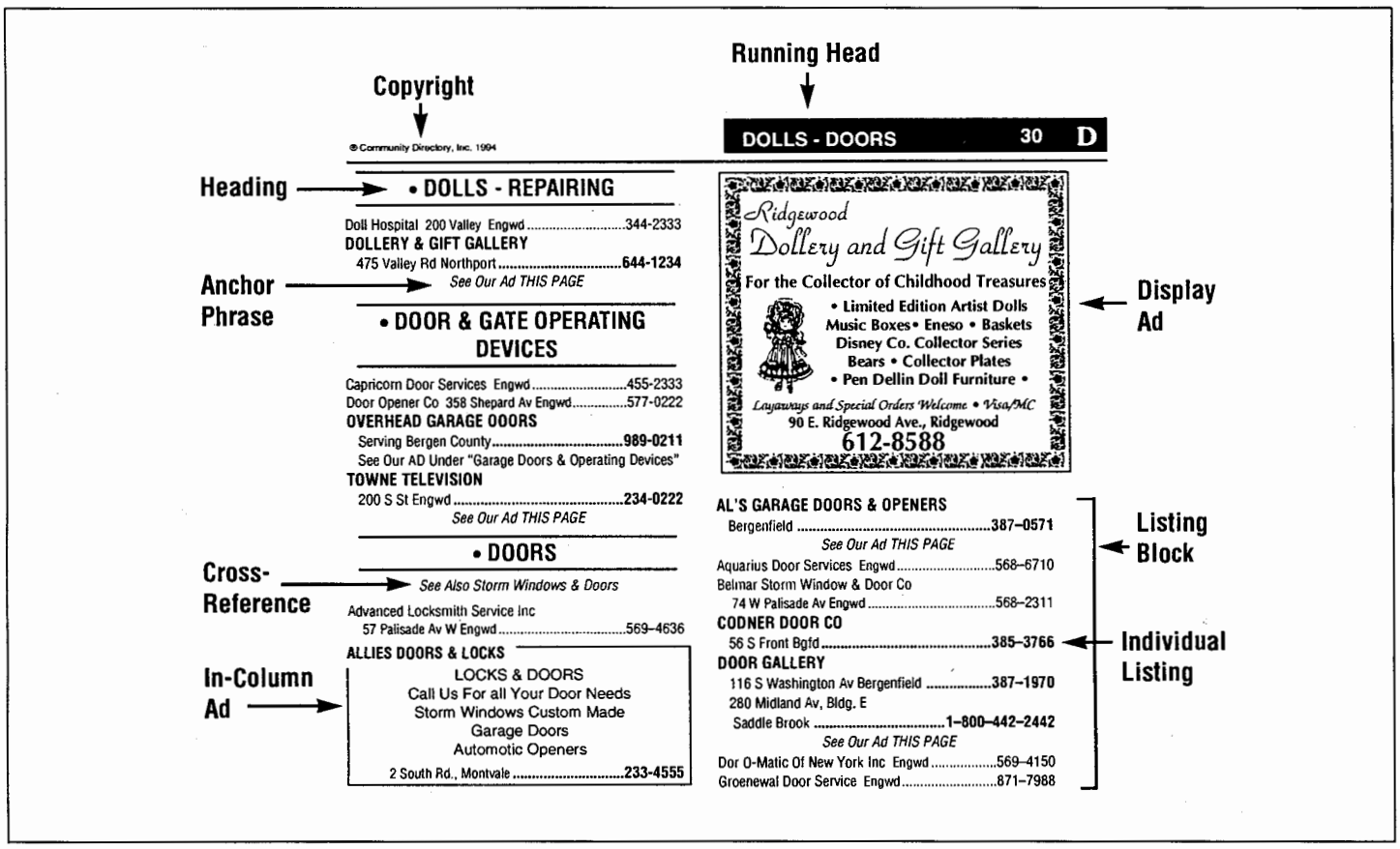

Figure 3. A typical Yellow Pages directory page, showing regions and objects.

Empty space left over on the page must be allocated "filler." Filler is typically community service or self-advertising text, selected at random from a filler library. Paging rules have significant consequences on the amount of empty space left over, and thus on the amount of filler needed.

A paging program must allow user intervention in any automated process. This requires that a graphical interface display an entire spread at a time and allow the user to make modifications. (This particular process cannot run on "dumb terminals.") Running heads, anchor page numbers, references, and the book's index (if any) must be maintained by both automatic and manual modifications to a page.

White Pages' Pagination and Typesetting

White Pages are considerably simpler because the display ad placement problem is eliminated, but many complex problems remain. *DIAD* generates complete White Pages in a single pass with no user intervention. These pages are generated in PostScript. Figure 4 shows an example of a White Pages directory page generated by *DIAD*.

PostScript is a page description programming language (with variables, procedures, parameter passing) that gives the programmer complete control of all white space in the area of a page where the printer can actually print. It also provides a level of physical device independence; the programmer need not be concerned with the printer or typesetter to be used, or with that device's resolution. PostScript supports scalability (increasing and decreasing the size of characters), rotation of text through any plane, control of leading (the amount of space between lines of text), and horizontal and vertical justification (the "averaging out" of white space between regions of print). PostScript also supports the mixture of text and graphics on the same page. *DIAD* generates PostScript by allowing the data in a users' M database (listings and ads) and tables controlled by the user (fonts, page parameters) to drive the compilation of a PostScript "object" for a particular job. Such an object includes both a program and data to be operated on by that program.

DIAD has used PostScript since 1985. For years, an objection in the industry was that PostScript devices were (and would continue to be) too slow. Now, the same industry insists on PostScript. The question of speed has been buried under a mountain of RISC processors and low-priced, high-resolution, large-memory printers. Note the parallel between M and PostScript as enduring standards that provide hardware and vendor independence to the implementor.

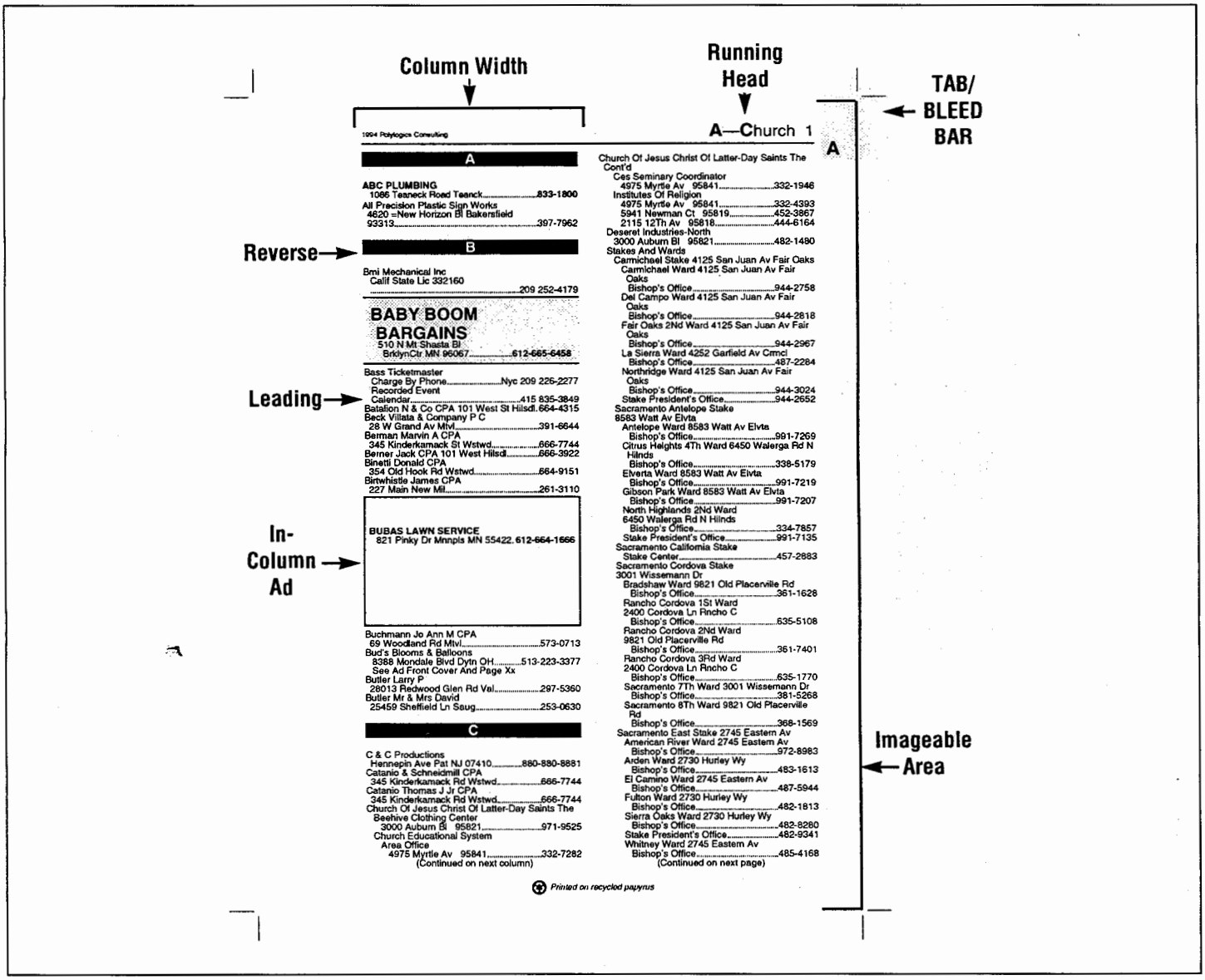

Figure 4. A sample White Pages directory page.

Summary

M has supported the growth of an increasingly complex and sophisticated application over a period of twelve years. The language's portability has facilitated the migration to multiple state-of-the-art platforms, and is credited by the vendor with enabling a commercial success in an industry not generally associated with M systems.

Mark A. Wieder is founder (1984) and president of Polylogics Consulting, Inc. He has worked in M since 1973, beginning with Digital Equipment Corporation's implementation for the PDP-15. Polylogics is developer and vendor of *DIAD*, an application supporting the telephone directory publishing industry. Polylogics also specializes in inter-dialect translation of M languages, most notably from nonstandard versions (especially MIIS) to standard M.