At the Creation: Chaos, Control, and Automation—Commercial Software Development for Archives

W. THEODORE DÜRR

The Information Revolution and the Software Developer

ONE DAY I ASKED A FRIEND who publishes in medical journals to describe an ideal computer system. Without hesitation he replied, "I'd like to sit in my study at home and do the following with a microcomputer: use my word processor; retrieve notes from an indexed database; transfer them right into my composition on the word processor; if necessary, dial up an electronic database, search it, and download some data into my computer; and tonight by computer send what I've written to a friend in Boston who will read it and send his comments back to me by electronic mail so I can read them tomorrow morning."

What does my friend have in common with archivists and librarians? He generates most of his own data, rarely visits libraries, and never darkens the doors of archives; even so, this modern scholar shares a community with librarians and archivists. It is the information community. Today many people are doing what my friend wants to do in his study; not quite as easily and in one place, but soon they will. Those who use microcomputers compose articles with word processing software by entering information. They retrieve information from notes indexed and stored electronically. They enter information from these notes (that is, fields in their databases) into compositions. They search external databases and can download information from them into stand-alone microcomputers. They send information from one computer to another.

W. Theodore Dürr is a Professor, University of Baltimore and University of Maryland and Chairman and a founder of CEO, AIRS, Inc., College Park, Maryland.

Information is not a buzzword that will fade tomorrow; it is the cornerstone of the 21st century. Libraries and archives collect and control information. Hierarchical control vocabularies which reflect conceptual relationships in the specialties have been developed. Libraries and archives, historically, became the center of the learning process and scholars came to them much as pilgrims journeyed to holy shrines. No longer.

The Information Environment

Today the flow of information has reversed. At a rapidly increasing pace, information flows outward from various sources to users. This does not obliterate archivists, curators, librarians, and records managers; it merely revolutionizes their jobs.

RLIN, OCLC, LEXIS, and Dialog are prototypes of this revolution. For the most part, however, these information systems currently represent institutions communicating with institutions. The direct communication envisioned by my friend is just around the corner. Information will come directly to users on floppy disks, CD ROM discs, or over the telephone wire. This is the environment for which the modern archives and the software developer plans.

In the world of hardware development there is an explosion. In the world of software development there is an implosion. Ten years ago colleague Adele Newburger and I decided that there would never be a commercial application for text database software because it required too much memory and storage. That was before the microchip! That was the last time we failed to bet on technology. Now hardly a month goes by that does not see the introduction of significant new technological advances giving us more for less—more storage, less space—more power, less cost. William Lowe, president of IBM's Entry Systems Division stated in 1986 that "more technology for microcomputers will emerge in the next five years than in the previous five...."[1]

While hardware vendors were fighting it out among themselves and producing one electronic miracle after another, some thought that software development might become an electronic cottage industry. A computer, a garage, and a programmer—robotic romance! Alas, it was not to be. We have entered the era of software implosion. Today consumers want software that integrates tables, text, and telecommunications. They want technology that is transparent and inexpensive.[2] The price of software is coming down as the utility rises. Systems which cost $20,000 and up on mainframes sell for $2,000 and less in the micro market. Few garage shops can afford to compete in such an environment. There probably will not be another Lotus or dBase story. Ashton-Tate, the producers of dBase, grew from a company deeply in debt with revenues of $3 million in 1981 to become a giant with sales of $200 million in 1986.

While the hardware and software worlds are exploding and imploding, respectively, the archival world is chaotic. New terms and acronymns abound, new demands are pressed, new functions are discovered, and new problems arise. One problem involves definition of some of our basic tasks.

At a Mid-Atlantic Region Archives Conference (MARAC) at Princeton in May 1986, William J. Joyce stated that archivists need to pay more attention to the "theory and functions of institutions. For it is in understanding bureaucratic institutional functions that we archivists achieve the strongest theoretical justification for our work." In other words, the chaos that surrounds us requires that we achieve both macro and micro views of our environment: the forest and the trees. Our practice must be redefined in light of changes in our culture. These changes are driven by technology—they are reflected in new institutional procedures and demanded by a fresh breed of scholars. Tomorrow's scholars, raised today in colleges where computers are as commonplace as typewriters, will expect information to be at their fingertips. Information will come in electronic forms such as facsimiles of documents and graphics received from all over the globe. The indexes to this material will be on disks that can fit into one's pocket and will be displayed on machines weighing under two pounds. If the opportunity is seized, we can create synergy out of the intellectual demands, the emerging technological utilities, and the new bureaucratic requirements.

Recent essays by archivists discuss time-honored subjects such as appraisal, preservation, and provenance, along with current issues such as relationships between automation and finding aids. Many of these articles are difficult to categorize as merely theoretical or practical because they both analyze from conceptual viewpoints and describe real world situations. Some examples are essays in *The American Archivist* by F. Gerald Ham, Harold T. Pinkett, Trudy Huskamp Peterson, Michael A. Lutzker, Richard H. Lytle, Frank G. Burke, Philip P. Mason, Meyer Fishbein, David Bearman, Charles Dollar, and Thomas Elton Brown.

While these archivists have been writing, another professional group has published articles on related topics. Known as information professionals, they write about records management, information retrieval, telecommunications, and theoretical topics such as the foundations of information science. From one of their publications, the *Journal of the American Society for Information Science,* we read essays of this type by Derek de Solla Price, Manfred Kochen, Paul G. Zurkowski, Lois F. Lunin, Edward John Kazlauskas, G. Salton, and E.A. Fox.

The archivists and information professionals have more than a little in common. All, in one way or another, are researching and writing about some aspect of the generation, organization, storage, retrieval, and dissemination of information. Whether the information is in manuscript or on a floppy disk, various social and technological characteristics of our time impinge on their work.

The first group, the archivists, are coming to grips with a fact of our time; as F. Gerald Ham put it, while the information revolution "gives us abundant information it creates an environment hazardous to its preservation."[3] With the coming of the microcomputer, Everyman has become not his own historian, as Carl Becker suggested; instead, as James O'Neill suggested, he has become his own records manager.[4] In 1986 I visited a company which has inspectors who fill out various forms. Beginning in 1988 they will not fill out paper forms, rather, they will write over a piece of glass, filling out blanks on the computer screen under the glass. This will automatically go into the computer and be indexed for every word. Thus the index becomes a creation of the individual who fills out the form—no one else is involved in the entry, transmission, or storing of the information, but anyone will be able to access and display the information on an IBM XT or AT computer. The second group, the information professionals, are coming to grips with problems of definition as they discuss basic terms such as "research," "information," and "science."

No archival software developer can ignore the contributions of either group. Both comprise the environment. Nor can users implement effective uses of systems software without following the rules of the archival and information disciplines. For purposes of this article, information science (I prefer the term "information discipline," however, "science" is universally used) is defined as the discipline that observes, experiments with, and defines the construction of automated systems which retrieve information generated and organized by bureaucracies for storage and distribution to selected audiences.

Information science helps us answer the questions "What are we doing?" "Why are we doing it?" "How are we doing it?" In the face of automation these are not idle questions. They are being addressed in various ways through research, publication, and conferences. An archival software developer asks: How do we control records in relation to managing records per se and in relation to retrieving them on the basis of their subject content?[5] The former approach concerns most records managers and archivists with an institutional responsibility; the latter concerns most historians, social scientists, and other researchers with information needs.

INFORMATION AND SOFTWARE DEVELOPMENT

Often, shopping for software is like buying a house. Buyers of a house designed by someone else either like it or they don't—it's too late to change it. Others prefer to hire an architect and have a house designed and built according to their specifications. Archivists are like those who demand an architect. This is true because of the absence of standardization. What one archive calls a record group, another calls a series; while one describes records only to the collection level, another describes them down to the folder, and so on. This means that each modern archivist wants his or her own custom-designed database management system.

The RLIN network is more like the mass-produced prebuilt house—archivists must fit into a predesigned format. That is, archivists must fit their information into the RLIN (MARC AMC) mold. Is there a way, asks the software developer, whereby archivists can build databases that match their own particularities but still be able to get the information out and enter it into the RLIN network when desirable? The issue here involves the MARC AMC format and data transportability.

Is there also a way to give users flexibility to design their own databases (by specifying field and record template characteristics) without requiring them to learn how to program? This is like having the privileges of the architect without the training. The issue here involves data entry and manipulation.

Is there a way, the software developer asks, to make sure that data, for example, an abstract, an ID code, or a full text, can be easily edited or appended and let the database administrator decide who has access? The issue here involves data integrity.

The software developer would also like to give people as many options as possible for searching databases. For instance, control vocabularies can be arranged alphabetically or hierarchically. Can the developer provide both options? Searching by proximity, truncation, or strings greatly facilitates certain textual needs.

There are two approaches to indexing: information retrieval and database management systems. Information retrieval (IR) software is found in packages such as Sci-Mate or InMagic. Its users are for the most part trained in bibliographic reference skills. They are familiar with mainframe database searches and serve as intermediaries between the information source (provider or vendor) and the information user. IR software grew out of the mainframe community and was usually placed in the hands of IR specialists.

Now, however, with the coming of micro power, flexibility, storage, and speed, database management, which was once the preserve of the mainframe environment, is becoming commonplace among desktop computer users. Recently a new term, "text based management system" (TBMS), has come into use.[6] This is replacing the term, data base management system (DBMS), because a TBMS has variable length fields (DBMS are fixed length), multiple value fields (DBMS are single), and produces record sorts and printouts which are more flexible than those in a DBMS. The TBMS type of software emulates mainframe DBMS functionality for text; the software packages MARCON, SIRE, and TEXTBANK attempt this, each taking different structural approaches to their program design and applications. Regardless of which of the above may be chosen, there is still a serious problem facing both developers and users. Many microcomputer desktop users are neither information specialists nor "computer jockeys." These users want a textbase management system that is specific to their needs and can be changed at any point to better serve their needs. Since a TBMS is generic, users need software that allows them to mold and shape what it does for them, without having to become software experts in the process.

Attempts to mold and shape a program are affected by the way people go about establishing a records control system. The problem is that no two people think alike. Therefore mismatches arise when the designers of a control system and its users interpret things differently. Baruch Fisshoff and Donald MacGregor distinguish between elite user groups who search a precise database (they use the older term) and nonelites who search an omnibus database which covers various sources. An example of the former would be cardiologists or fighter pilots, each with a relating homogeneous perspective. An example of the latter would be users of Dow Jones News Retrieval or ERIC. The latter group is heterogeneous. Thus archival database design, which of necessity includes a wide array of source material (to be controlled) and, potentially, a wide array of users (to be served), must do two things. For the "elite" users, for example, who know the provenance of a collection, a system must offer very precise information related to offices, services, dates, and so forth. For the "omnibus" users, the system might offer more broad information related to subject. One system may encompass textbases that reflect both approaches (one elite; the other omnibus) and a good TBMS would allow simultaneous searches across both.[7]

The software developer therefore tries to provide a TBMS package that is as generic as possible. But that is not enough. The administrator who builds the textbase must think through the issues involved, the applications (end result) desired, and the expectations of potential users. Careful planning at the time of textbase design is a sine qua non for a good TBMS, along with clear instructions to users once the TBMS is completed.[8]

To understand software archival TBMS design, we may think of a three-tiered universe (see fig. 1). The tiers are (1) The Repository Management System, (2) The Software Management System, and (3) The User Management System Requirements.

Within each tier are four layers; each layer represents the same concept in each tier. The three tiers (and the layers within them) are interactive. These layers are: *A*—Objective (the purpose), *B*—Program (the input and throughput or how the purpose is achieved), *C*—Result (output or what happens), and *D*—Interface (expressed in each module as *Information*). Information is the key intellectual property, the common denominator, which is used by people involved in various roles in all three tiers. Management of information is the key to what professionals do—they manage information about records in order to manage records.

The top layer (*A*) of each tier is labeled Objective; *capture, control,* and *use,* are the key terms throughout in this level. The activity at this stage involves capturing information about the context and content of records for control of the records so that they may be used whenever necessary.

Level *B* in each tier is labeled Program; *description, repository requirements,* and *software applications* are the key terms in this level. This involves meeting the repository's requirements so that description of the records will be accurate in the software applications; it further entails recognition that adequate information about a record system includes not only the subject content of the record but also information about the record's context. Since the origin and/or "life cycle" (use over time) of records is important to their custodians, provenance related information is required. At the same time there often will be requests that are subject based. For this kind of query the user wants information

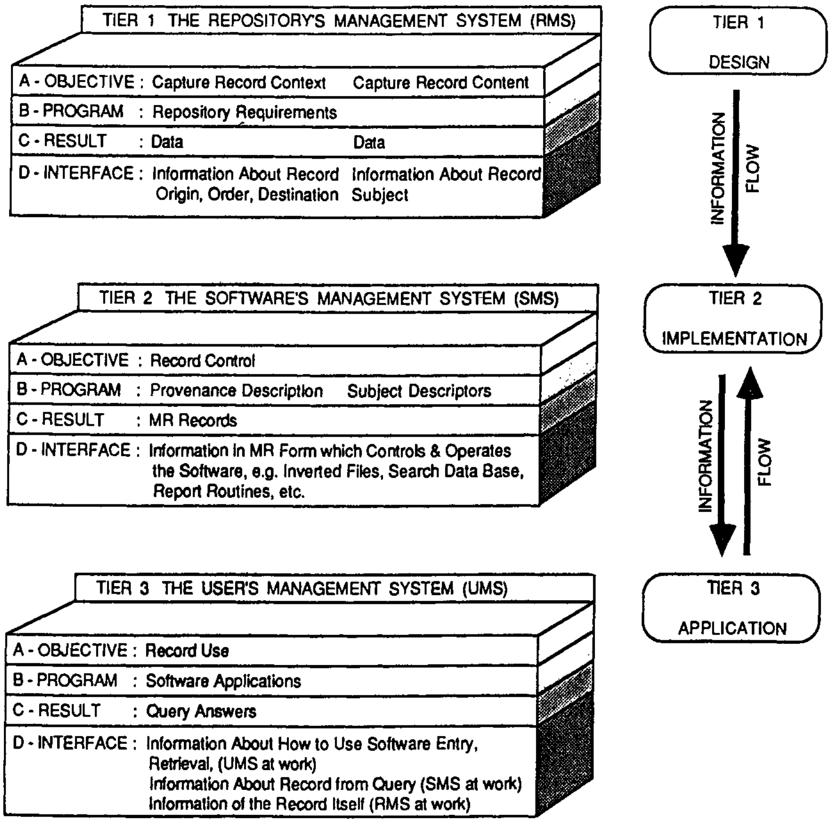
TIER 1 THE REPOSITORY'S MANAGEMENT SYSTEM (RMS)
A - OBJECTIVE : Capture Record Context Capture Record Content
B - PROGRAM : Repository Requirements
C - RESULT : Data Data
D - INTERFACE : Information About Record Origin, Order, Destination Information About Record Subject
TIER 2 THE SOFTWARE'S MANAGEMENT SYSTEM (SMS)
A - OBJECTIVE : Record Control
B - PROGRAM : Provenance Description Subject Descriptors
C - RESULT : MR Records
D - INTERFACE : Information in MR Form which Controls & Operates the Software, e.g. Inverted Files, Search Data Base, Report Routines, etc.
TIER 3 THE USER'S MANAGEMENT SYSTEM (UMS)
A - OBJECTIVE : Record Use
B - PROGRAM : Software Applications
C - RESULT : Query Answers
D - INTERFACE : Information About How to Use Software Entry, Retrieval, (UMS at work) Information About Record from Query (SMS at work) Information of the Record Itself (RMS at work)
TIER 1 DESIGN
INFORMATION FLOW
TIER 2 IMPLEMENTATION
INFORMATION FLOW
TIER 3 APPLICATION

Figure 1. The three tiered universe of the text based management system.

about a given subject in all extant records. Thus the ideal system (at level *B*) describes records in terms of both provenance and subject content.

Level *C* in each tier is labeled Result; *data, machine readable records,* and *query answers* are the key terms in this level. This involves data which the repository holds (as a result of accession/accretion/collection). It is enhanced by machine readable (MR) records about the data. When the required information has been entered about the context (provenance) and content (subject) of the record, a textbase[9] exists. Data in the computer at level *C* can be searched. This makes it possible for query answers. The term *data* is used here because the information may be in the form of graphics rather than text, in which case data is a broader term.

Level *D* in each tier is labeled Interface; *information* is the key word. In tier 1 this involves information about record origin, order, destination, and subject so that the MR records in tier 2 may be searched for information related to the records. Thus, information—about how to use the software, about records that meet the definition of queries, and of or about the archival records themselves—may be retrieved. An adequate retrieval system implemented with the aid of automation ultimately requires that all things be broken down to common elements.

Level *D* is named interface for a very precise reason. The outstanding *Time Life* series on the computer defines interface as "electronic circuitry that allows two devices to communicate with each other;"[10] the phrase "two devices to communicate with each other" is the heart of the definition. If the word *entities* (which could refer to the bureaucracy of a repository, to the computer system, or to the user) is substituted for devices in the definition, then the interface emerges as the critical component that makes everything else work. Interfaces allow one or more entities to communicate with each other. For example, imagine a conference call to 1000 participants; AT&T long lines and related equipment function as the interface. The messages communicated as people speak are the information! In our three-tier example, information is the message. In tiers 2 and 3 this information/message is put into electronic circuitry so that repository information may be communicated with facility and precision to the user.

To summarize, layer *A* represents the three categories that must be considered when planning an archival information system. Layer *B* suggests definition which outlines the parameters of a program. These two layers in tiers 1 and 2 do not directly involve the computer. The next two layers, however, are impossible without the computer. Layer *C* represents the actual construction of the software program, its functionality, size, operational methods, and so forth. Level *D* is the interface that resides in the computer. The interface pulls everything together according to the way both the database creator and end user want it.

Metaphorically, a database or textbase is an electronic garden of Eden. It can be a creation of beauty and power. Just as Adam named the animals, the modern textbase administrator names the fields and values in them; the modern researcher calls for whatever combinations of creatures are desired from the textbase. In seconds they appear on the screen.

The text software developer's task is to provide this power as fully as possible. The developer seeks to build software which allows each administrator to build a database that offers as much flexibility and power as possible for both the elite and the omnibus user. The key terms are: chaos, creation, control.

OF MICE AND TRAPS IN SOFTWARE DEVELOPMENT

The MIT Enterprise Forum has chapters in Boston, New York, and Baltimore-Washington. Financiers and people interested in technology attend the meetings where start-up "hi tech" companies present their technology and business plans. At one of the meetings in 1984 an entrepreneur presented an electronic mouse that made it easier to move the cursor around the monitor screen. The mouse was not a new product but this businessman thought his was better.

Members of the forum warned him of the traps that await all small businesses. The first trap involves the market. A successful business needs a market that wants its product or service. The market has to be large enough for the business to make a profit—good ideas and profits do not have a one-to-one relationship. Broadly speaking, there are two kinds of markets—horizontal and vertical. The former has many customers. In the world of dentistry this would be toothpaste, not braces; in the world of software this would be word processing, not RLIN. Firms that market horizontally seek a broad customer base and price their products accordingly. The software developer can create a ratio which compares product development cost and item sale price; the sale price is lower in relation to the development cost. In the vertical market it can be higher because volume will be lower, but if it is too high, market resistance and competition will defeat sales efforts.

The vertical market is often called the niche market. In the art world this is the market of autographs, not reprints; in software this is the market of specific applications. SPINDEX and NARS A-1 software were developed specifically for archival application. Yet no matter how valuable the applications are, the software developer must create a product that is flexible, easy to use, and inexpensive.

Flexible	*Easy to Use*	*Inexpensive*
Allows user to define fields, database, and search strategies	Operates in natural language (English)	Price is competitive
Offers various means of entry (keyboard, OCR, electromagnetic) and retrieval (full text, string searching, Boolean connectors)	Provides attractive, easy-to-read screens and clear instructions	Updates cost much less than original version
Generates a variety of report formats	Operates by means of simple machine commands	

The second trap involves competition. If a competing, inferior product is already for sale, or even if it is introduced at the same time but advertised more widely and effectively, the company with the inferior product will retain the market. Competition can be overt or subtle. The overt approach is when a developer tries to blast the opposition with a cheaper price, accompanied by either a sophisticated advertising campaign that may promise more than the product delivers or a cheaper price accompanied by more functionality. A subtle form of competition is when one product is able to attract funds for support and is backed to the exclusion of competing products. Probably the best form of competition is to put the product into the hands of competent reviewers. Smart software buyers know where to look for good reviews of products and reviewers' comments influence their decisions.

The third trap concerns operations. This is especially critical in software development. The software developer has to maintain research and development, production, distribution, quality control, customer support, and internal administration. Since omniscience is not bestowed on software developers any more than other mortals, frequent communication with intelligent system users is necessary to ensure that bugs are weeded out and improvements are woven in. To maintain this exchange, software developers have alpha and beta sites. An alpha site is one that uses versions of the software while it is under development. This is extremely valuable. Design modifications, manual changes, and other adjustments can be integrated as a result of real world experience at the alpha site. Beta sites are installations that use the software after it is finished but before it is released to the general public.

Bugs, the perpetual pests of the software industry, are usually discovered at beta sites. While software developers test as rigorously as possible, they cannot imagine all the stresses and strains real world users will place upon their product. There is no mass-produced commercial software that never had a bug. Every time a modularly designed software program is changed, it has to be recompiled, causing the possible introduction of new bugs. Therefore, a rigorous testing protocol is necessary; at the same time, there is pressure to start shipping the new version.

Testing, evaluation, and modification, are continual efforts in the software industry. This is directly related to customer support and software improvement. Users continually offer suggestions for improvements. The trick is to recognize potentially marketable suggestions and pinpoint technological breakthroughs that can facilitate viable new enhancements.

The fourth trap involves finance. There is no hope for a firm whose sole market is archival repositories. The total market potential among archives is about 3,000 repositories. If the product is software, the customer base, in 1987, is cut in half because many archives are not ready to automate. Since the lead time—that is the time it takes a customer to decide after being introduced to the product—is between six months to one year, the potential customer base for the first year reasonably comes out to about 750. If the competition factor is added, the potential is advisedly narrowed to 375. If the product sells for $1,000, and requires $4,000 worth of hardware and related equipment, the number might again be reduced by half to 188 or $188,000 in sales. Since the software will cost at least $300,000 to develop and support (not to mention advertising and distribution), no one would take such a risk. The software developer must have a larger agenda, a bigger market in mind.

First, the developer may "add value" to an existing mass distributed product. For instance, if the developer's software can make WordPerfect function better by automatically indexing text then users of this widely distributed product may buy the software, and WordPerfect may distribute it. This puts the sales organization of a major software firm to work for the small developer. This is known as a VAR (Value Added Resellor) relationship. Another tactic is to have the software "bundled" into another product. For instance, if a producer and marketer of a CD-ROM database requires a text filer and indexer, the developer can execute a license agreement with the company selling the database and reap royalties on every sale. This is known as an OEM (Original Equipment

Manufacturer) relationship. These and similar techniques will broaden the market base and may lead to survival.

The archival software developer has another problem which will not be solved without CD ROM (Compact Disk-Read Only Memory). When information is communicated on paper (excluding graphics) it is either in numeric or script symbols. In the world of automation there is a vast difference between the two. For example, the number 5,465,025,015 requires 13 bytes of storage in its numeric form. On a double-sided double-density disk which has room for 360,000 bytes, this is not much. On the other hand, 5,465,025,015 words (averaging 6 letters each) would require 38,465,115,105 bytes of storage or 106,264 floppy disks! This explains why, before the invention of the microchip, most databases were numeric.

Only with the chip have text databases become feasible. A megabyte is one million bytes. Microcomputers with 80 megabytes of storage are common today and computers with over 100 megabytes of storage are not uncommon. When large volumes of text are stored, an index becomes necessary. This adds to the storage problem because the index, too, takes up space. While a hard disk of 30 to 60 megabytes will often suffice, the new CD ROM technology most adequately addresses the problem. CD ROM is similar to the small, compact or digital disc used in the music record industry. As an archival device, one 5¼" CD ROM disc will hold 500 megabytes. An entire encyclopedia will fit on a 12½" disc holding one gigabyte (one billion bytes). Obviously indexing software is a sine qua non for using this much data.

Solving Software Dilemmas

The software developer can do all of the above and more. Decisions, however, involve tradeoffs. The more options a software program offers, the greater its size. The larger the program, the more space it requires. The more space, the longer the search, or, if the search time is to be shortened by certain techniques, then the indexing time is increased. Software is like putty. If you push it a little here, it comes out a little there. Software can do just about anything given space and time. The tradeoffs are between function, time, and space.

Building good software and databases is an art. For that art to be fully appreciated, the product must be used correctly. As the demand for text- and records-control software grows, many customers will be neither information scientists, reference librarians, nor archivists. The librarian and the archivist, furthermore, may know little about database construction. Inappropriate uses of the software may result. A good manual and tutorial are often not enough; the solution resides in workshops and seminars. It is worthwhile for the purchaser to invest a little more time and money in order to learn to use the product correctly and it is worthwhile to the developer to make the cost as reasonable as possible.[11]

Most software firms have long lists of ideas and short lists of resources. The chief of technology and development for one software house always says that he will deliver anything given time and money. When creating a budget, the developer must remember that programmers tend to underestimate the time required to complete a project and marketers tend to overestimate sales projections.

Whatever the business strategy and the survival techniques, the small firm has to back up its product, deliver on its promises, and manage its resources well. The developer may have a dream which envisions thousands of people using the firm's software but the dream must not cloud the developer's eye to hundreds of details which have to be managed on a daily basis. At the same time, the details cannot be allowed to decimate the dream.

References

1. Lowe, William. *Infoworld,* 28 May 1986, p. 6.

2. In May 1986, the WordPerfect corporation published a "Library," an integrating tool box which locates programs in various subdirectories and with a "Notebook list management program," allows users to move data around via a "clipboard." Another product, MacPik, is a utility that inputs files of one type and outputs them as other types, allowing communication with downloaded files. See Lowe, William. *Infoworld* 8(7 July 1986):32, 37.

3. Ham, F. Gerald. "Archival Strategies for the Post Custodial Era." *American Archivist* 44(Summer 1981):209. We probably do not want to store so much information. Knowledge, which is information in context, is what we need to store. Learning to differentiate between the two is one of the tasks ahead.

4. O'Neill, James. "Archives in the Eighties." Unpublished paper presented at the National Association of State Archives and Records Administrations, 31 July 1980.

5. During the last 8 years there has been much discussion about records management and indexing the information content of records. See Lytle, Richard. "Intellectual Access to Archives: I. Provenance and Content Indexing Methods of Subject Retrieval." *American Archivist* 43(Spring 1980):191-206. The results of the experiment were first reported in Lytle's 1979 doctoral dissertation "Subject Retrieval in Archives: A Comparison of the Provenance and Content Indexing Methods," at the University of Maryland. See also Dürr, W. Theodore. "Some Thoughts and Designs about Archives and Automation, 1984." *American Archivist* 47(Summer 1984):271-30.

6. See Puglia, Vincent. "TBMS: Database Power Unleashed." *PC Magazine* 5(25 Nov. 1986):211-30.

7. Fisshoff, Baruch, and MacGregor, Donald. "Calibrating Databases." *JASIS* 37(July 1986):222-23.

8. Work in this context leads, ultimately, to artificial intelligence. Key terms for searches are: rule based systems, knowledge bases, decision support systems, expert systems. The author assumes no responsibility for those who, outward bound, get lost while trying to follow the trail.

9. The two words "data" and "base" were standardized to one word "database" in the *Annual Review of Information Science and Technology* (ARIST) in 1979. The same, presumably, will happen to "text" and "base."

10. *Input-Output.* Alexandria, Va.: Time Life, 1986.

11. Many software purchasers who want the functionality of a database do not understand its makeup. Thus training is often necessary. A good primer is: Humphrey, Susanne M., and Biagio, John Melloni. *Databases: A Primer for Retrieving Information by Computer.* Englewood Cliffs, N.J.: Prentice-Hall, 1986.

This Page Intentionally Left Blank