ARCHIVES AND MUSEUM INFORMATICS

TECHNICAL REPORT

ISSN 1042-1459 No.2

# Collecting Software: A New Challenge for Archives & Museums

by David Bearman

Originally published as Archival Informatics Technical Report
vol.1, #2, Summer 1987
Reprinted 1990

# COLLECTING SOFTWARE:
# A NEW CHALLENGE FOR ARCHIVES & MUSEUMS

## EXECUTIVE SUMMARY

For forty years software has been an important creative product of our society. Its intellectual, social, economic and political impact has shaped the contemporary world, yet the community of culture preserving institutions has failed to document its evolution. There is not a single archive or museum in the world devoted to software and no substantial collecting of software history has taken place in other repositories although software is being written every day which defines the way in which we work. Bureaucracies, including governments, are entirely dependent upon software to faithfully execute the policies (including laws and regulations) which they have established, yet archives, those gaurdians of bureaucratic accountability, don't retain software. Popular culture and the arts have both been transformed by software, but museums have yet to collect it. To archives and museums, software is still alien and insubstantial.

This report examines the history of software and its influences on our society and it addresses the barriers to collecting software as a cultural record. It identifies essential policy distinctions which administrators will need to consider between software collections and other collections of archives and museums. It examines the ways in which software can best be described, made available to researchers, and exhibited and it proposes a framework for a descriptive vocabulary. Further, it identifies the physical requirements and management issues associated with the retention and storage, retrieval and use, of software in cultural repositories.

An earlier draft of this report was prepared for the Computer Museum in Boston as the framework for a discussion with staff of the Smithsonian Institution and the Charles Babbage Institute on establishing a national software collecting consortium. The report, therefore, considers approaches to multi-institutional collecting issues such as documentation strategies and collection policies, cooperative acquisitions and information sharing.

The body of the report addresses five fundamental questions about such a program:

What is the domain of a software archive?
What is the mission of such a program?
What policies are required to implement a software archive?
Who are its users and what are its uses?
What procedures are needed to establish a software archive?

# COLLECTING SOFTWARE
# TABLE OF CONTENTS:

## INTRODUCTION

The digital computer, an electronic device capable of carrying out a sequence of instructions communicated to it in a "program" which can be executed without further human intervention, had its origins in wartime applications in the 1940's. It emerged from the war as an experimental machine, evolving rapidly at universities during the late 1940's into a practical engine for numerical calculations and logical operations. Its impact on our daily lives has been so dramatic during the last thirty years that few other events in human history can be appropriately compared; and it is much too early for us to know how it may transform our civilization.

However complex and interesting the machines which have launched this revolution are, it is not they, but the instructions people have written for them to execute which are redefining the world in which we live. These instructions, or programs, written in languages which can be compiled or interpreted into machine code, are among the most exciting creative achievements if our day, yet their content, structure, cultural impact and socio-economic significance are only now becoming the subject of critical historical study. Early software masterpieces, monumental intellectual achievements of the genre, have already been lost to future study through neglect. And contemporary software concepts, rarely first embodied in widely sold general purpose systems, will likewise be lost unless the cultural history repositories of our day - the museums and archives of our contemporary society - take action soon to prevent the disappearance of this record.

In 1986, the Computer Museum, a cultural repository dedicated to collecting computers and computing artifacts, decided to explore the feasibility of a software archive. They contracted with the author to examine the nature of software and its documentation, and define what it meant to collect software as a cultural achievement. This document grew out of that project. It calls a collection of software and software related documentation a "software archive" and examines what a software archive might be. It endeavors to demonstrate not only that such a program is possible, but also that it would have substantial cultural value, and may even be necessary, to document modern political and economic institutions. It suggests that most existing museums and archives need to consider software within the scope of their

present collecting policies.

This report poses broad questions about software, about documentation efforts and about the requirements of such a program. It shows that there are no fundamental barriers to proceeding and then identifies guidelines which could be used in administering such a program.

## I. THE DOMAIN OF A SOFTWARE COLLECTION

To plan for the preservation of software, we must first understand what it is and what social activity generated it. We must then examine the types of evidence these activities left behind. Finally we must consider the challenges of collecting and providing access to an adequate record of that activity. We may make the history of software itself our framework for collecting or we may take an external reference point and make the history of that area of human endeavor, an "application arena", our framework. In either case we must understand the nature of software itself.

### A. SOFTWARE HISTORY AS A FRAMEWORK FOR COLLECTING

Computers are machines operated by instructions which are input to them in a form they can execute. These instructions, or programs, began to be called software relatively late in the history of the computer, to distinguish them as components of the system from the hardware, or equipment.[1] This distinction, while sufficiently clear in principle, has in fact never been absolute; the emergence of the term firmware, to denote instruction sets embedded in hardware, reflects the increasing resolution of this still gray area. For the puposes of a survey of the universe of potential software documentation issues, software must be considered to include any instruction set, however embodied, thereby forcing us to consider (although not necessarily to be subsequently collect) firmware by the original equipment manufacturer and by third parties.

Likewise, for the purposes of assessing the range of a comprehensive record of software, no purpose is served by defining an arbitrary moment

[1]My search for the origin of the term "software" ended at the Time-Life volume entitled Software which locates it as being in general use by the early 1960's. I am willing to assume that if there were a better answer, Time-Life researchers would have found it.

after which computer programs can be considered software; the potential domain for a software archive must extend to the earliest computing devices. Experimental computers, those developed prior to 1948, lacked control units and stored logic sequences and operated in part under the control of mechanically set switches. Although "programs" were written for these machines,[2] the processing was not fully controlled by the programs. Computers for which software could operate as a complete instruction set, required transfer of control and stored logic (demonstrated by the SSEC machine, 27 Jan 1948), electronic registers (demonstrated by the MADM, June 1949), a control unit and a memory size adequate to support it (demonstrated by the ACE machine in 1950, and EDVAR in 1951), and paging of memory (although not automatic, demonstrated by EDSAC 6 May 1949). All these features did not come together in practical machines until the advent of the UNIVAC I and IBM 701, the first "mass produced" computers[3], in 1952/3. Nevertheless, a number of quite important computer programs were written before 1952/3, and many important software concepts emerged from this period. In documenting the history of software, we cannot ignore these developments simply becaust the term "software" had not been coined, or because executing a particular program on these machines required a specific prior physical configuration and, sometimes, human intervention during execution.

But the symbiosis in these years between particular devices and the specific programs written to control them does have implications for the establishment of software archives. Since each of these experimental computers was unique and the methods of expressing instructions were non-

---

[2]Much important software developed on pre-1946 machines is known from the published literature, although this is inadequate for understanding the sources of certain ideas or the trial and errror involved; cf.
Adele Goldfine's, 1946 trajectory calculation program for the ENIAC which have recently been reprinted in Randell, Brian, ed.; The Origin of Digital Computers: Selected Papers, 1982
Claude Shannon's, 1938 logic program for his switching relay published in Transactions of the American Institute of Electical Engineers, or R.E. Beard's, 1942 actuarial table calculation programs published in the Journal of the Institute of Actuaries, v.71 p.193-227

[3] Mass produced is to be understood as a method of production, not an indication of volume. UNIVAC sold only fifteen of its Model 1's. IBM manufactured only 19 of its model 701 (the number sold is debated) and sold only fifteen of its Model 702.

standard, and tied closely to the physical device, these programs and the hardware on which they ran must be considered to be a piece,with responsibility for documenting both assumed by the cultural repository which acquires the hardware. The function of a software archive with respect to such early programs will be to impress upon the museum repositories that in undertaking to preserve the meaning of the artifact, they are assuming responsibility for documenting the problems which the machine was given, the way in which these problems were set, both in hardware and software, and how execution took place. There may be occasions when the machine for which such an early program was written no longer exists or is not being preserved, and, if adequate documentation exists to describe the machine itself and its functioning, the software written for it may be intelligible. With few exceptions, however, the domain of a software archive will encompass computer programs written since the mid-1950's for machines (and later operating systems and languages) which are non-unique; software for unique devices will continue to be collected along with those artifacts, by museum curators.

Software may be the product of the machine manufacturer, academics, the user, or of commercial "third parties" (who are relative late comers). Writing computer programs as an activity separate from designing and building computers, arose with the commercial distribution of computers in the early 1950's. However, when it first moved out of academic laboratories, it remained for several years largely confined to the province of the computer manufacturers, in collaboration with their (large) customers. An early example of a program of this sort was that written by UNIVAC engineers as part of the first computer sales effort - the splashy joint venture with CBS in November 1952 to predict the Presidential election results from exit polls and to launch the UNIVAC I. Collaborative relations between the customers of computers and the manufacturers, with spillover between hardware design and application, continued to be the norm in the period when each computer was dedicated to a single task. Thus IBM developed not only the software to run the Social Security system on its model 701 in 1953, but also developed (and subsequently sold) a tape processor to store the dataset.[4]

---

[4] The IBM tape unit is discussed in Rene Moreau, The Computer Comes of Age: The People, the Hardware and the Software (Cambridge MA, MIT Press, 1984). Also, see

As with many government procurements of the period, new hardware and software concepts were developed around the statement of a practical problem and multi-million dollar computers were sold to process a single application. Because each model of commercial computer during this period was sold to a small number of sites (rarely over 20), the documentation of software and hardware after the mid-1950's can be segregated. In documenting custom application systems, both the records of the design process and those of the client are important. Unfortunately for the future of a software archive, it does not appear that either the size of these early custom systems or their huge costs assured the preservation of a documentary record. Preliminary inquiries have failed to find records of several important developments.

An explosion of computer programming which laid the foundations for most concepts being refined today, took place in the late 1950's. With the development of paging (moving data between layers of the heirarchy in fixed blocks) and the evolution of drum memories and eventually disks packs (introduced by IBM as RAMAC in 1956), the computer became a sufficiently general tool to require local engineers who were capable of writing instuctions for new applications. Many of these locally developed tools were adopted at other sites either informally, like Bob Patrick's (General Motors) "monitor system" for the IBM 704 or formally, like Ray Nutt's (United Airlines) assembler for the IBM 701.[5] Such applications in themselves, together with the development of general languages for writing machine independent insructions, such as FORTRAN, LISP, ALGOL and COBOL, all of which had their origins in the late 1950's,[6] ushered in a new phase of computer program evolution, during which the concept of software as a separate entity begins to have meaning. Because a single model was available in numerous sites (sales now exceeded 1000 for successful computers),

---

Charles J. Bache, Lyle R. Johnson, John H. Palmer & Emerson Pugh, IBM's Early Computers, (Cambrige MA, MIT Press, 1986).

[5]On Patrick see Moreau, ibid.; on Nutt, see Bache, ibid.

[6]John Backus "History of Fortran I, II, III" , in Wexelblat, ed. History of Programming Languages, (NY, Academic Press, 1981); also articles on LISP, ALGOL, COBOL etc.

and because numerous applications were being run on a single machine, expertise was developed in house, and software concepts began to be developed from a community of programmers, rather than as a reflection of the manufacturers' requirements for installing a system. Because the control unit was given increasingly complex tasks to manage, with the advent of new input and output devices and deeper layers of storage, the need for operating systems was general. And because operating systems could handle physical level instructions which were repetitive, higher level languages began to evolve, each requiring a compiler for every new machine, and each giving rise to application routines and libraries in a community of specialists. By the late 1950's this activity was reflected in the evolution of institutionalized software exchanges, such as the IBM user group, SHARE.[7] Documenting the evolution of software will involve documenting these social contexts for the dissemination of knowledge of operating systems and libraries of system management routines in the transition from single processor systems to multi-processor sites with numerous I/O devices and complex communications.[8]

During the 1950's, computer programming became evolutionary in another extremely important respect - programmers built higher level concepts on lower level ones using new computing "languages". Many important programming concepts of this period were embodied in languages, so that each language came to be regarded as being most appropriate to particular kinds of problems and each provided only certain tools. Thus the history of programming languages is particularly important at this time; fortunately languages are the one area in the history of software for which a modest archive has already been established.[9] The emergence of programming

---

[7]Bache, op.cit. #4 discusses the origin of SHARE. It is worth noting that IBM archives contains "most of the published SHARE reports."

[8]Brian Kahin discusses the sociology of software distribution and the efforts to establish a new mechanism better suited to the needs of academia, in his draft report on the EDUCOM Software Initiative (November 11,1986), cited by permission of the author with the understanding that a final draft of this report will be issued by EDUCOM in the near future.

[9]Jean Sammett maintains an archive relating to languages at the IBM Federal Systems Division, in Bethesda Md.

languages marks a radical departure with major impacts for any potential software archive; programming languages are documented virtual machines and software written in them can be comprehended without the necessity of maintaining the kind of machine specific documentation required to understand software written in lower level code. The history of LISP, which was developed before John McCarthy had access to a computer which could run it, and of the FORTRAN programs written by prospective IBM 704 customers before a compiler for FORTRAN was written, provide a dramatic illustration of this independence of such virtual machines from actual computers.[10]

While operating systems and programming languages were being constructed as the base of a software revolution, computer applications were attracting substantial public attention. Following the publication of Edmund Berkeley's Giant Brains: or Machines that Think, in 1949, such journals as the Harvard Business Review were quick to focus on the potential of computers, and devoted numerous serious articles to following the utility of applications in their areas of interest.[11] Even though general purpose computing was still a modest component of the usage of computers (by far the most extensive use of computers was in military applications such as atomic energy, ballistics and eventually the space race), the potential of software as a competitive edge was clearly demonstrated by some pioneering efforts, such as the SABRE airline reservation system developed by American Airlines and IBM from 1956 to 1962. This developing public awareness of software, and of the ways in which software could be used to competitive advantage, is itself an object of documentary interest in the domain of a software archive.

While software was already a commercial asset in the early 1960's, as demonstrated by the success of Computer Sciences Corporation which was

---

[10] John McCarthy reports that since the N.E. Computation Center was not scheduled to get its IBM 704 until 1957, the design of LISP, including such central issues as how to use the 15 bit register which gave us cdr and car, were made without a computer. In the history of FORTRAN, we find sites which were awaiting the delivery of the compiler writing application code to be run when a compiler was delivered. See Wexelblat, op.cit. #6

[11] Edmund C. Berkeley, Giant Brains: or Machines that Think (NY, John Wiley & Sons, 1949). also; Editors of the Harvard Business Review, The Digital Computer: Monster or Slave (Boston, HBR, 1955)

formed in 1959 to sell contract-developed programs, the emergence of an independent software marketplace for software as a commodity required a sufficiently large population of computers of a common type to assure that multiple users would purchase or license a given product. This condition, was on the verge of being met by the IBM 1400 series when IBM pulled the rug from under its users, destroying their software investments with the introduction of the IBM 360. It was met by the 360 and its successors. Operating systems were sold as part of the computers they operated; it was no longer possible for local programmers to make or even borrow code since the number of lines lines of instructions required had grown from a modest 5,000 lines for the IBM 650 to several million lines for the IBM 3601[12] Other software tools, as they began to be developed, were leased by the manufacturers. While application software was being developed in numerous computer using organizations, it was not until the 1970's that it was commercially distributed by third parties dedicated to developing software for resale.

By 1970, one can begin to talk about a fourth period in the the history of software - the period of commercial, third party, unbundled, software development and the parallel development of a distribution system which can only exist in contast to the commercial system, that of public domain or free software exchanges. Groups of organizations with common requirements whether trade organizations, government agencies, or non-profit cultural institutions, began to develop software in consortia or developed mechan-isms to exchange software among themselves. The emergence of software as a commodity and the creation of pricing structures, markets and suppliers, is a part of the domain of a software archive.

Computers were developed to support military objectives and perform calculations at a speed which surpassed that of people, but it was not long before the computer was put to work as a routine file clerk in civilian projects, maintaining large sets of data and retrieving required information

---

[12] John Pfeiffer, The Thinking Machine (Philadelphia, J.P. Lippincott, 1962) based in large part on a TV program broadcast by CBS, oct. 26 1960, as part of the Centennial of MIT. In retrospect, given the speed at which innovations began to impact on daily life, it is startling to realize that the MIT Computation Center, the earliest computing facility at a univeristy which was not dedicated to developing new computers but using existing computers, was opened to researchers from 30 universities in June of 1957.

from them. In these functions the computer performed work which could have been given to people, but was repetitive and dull. Routine bookkeeping on a large scale was error prone; searching for files of cases was likely to result in misfiling them after the tasks were completed. The computer not only assisted in these operations, it eventually made possible file management on a vast new scale, such as that of the credit card companies or the Social Security Administration, thus shaping the way in which business was conducted by making it possible to conduct business in that way.

Until about 1970 the impact of computers on such large scale enterprises was imitative of the human processes (although, because computers cost over a million a dollars and were expensive to lease and because large staffs and special facilities were required to run them, we can assume it was cost effective). During the 1970's this changed. As computers came down in price and their performance was improved more and more processes were automated, thereby qualitatively transforming the functions which they controlled. By the end of the 1970's, computers as powerful as those used by large scale enterprises in the 1950's were available to small businesses, and they began changing the way in which they managed their internal functions. Concurrently, the larger enterprises had figured out how to coordinate a great deal of their processing from the initial component order to the final collection of receipts on the product, and this end to end automation was beginning to involve the ultimate customer, the consumer. The consumer became aware of automated inventory systems at the cash register, of automated banking transactions at the 24 hour teller machine and of large databases about himself through the "personalized" targetted mailings he received daily. The domain of a software archive encompasses documentation of such fundamental cultural effects as those of changing the scale of business and government, transforming mans sense of self[13], and altering the balance of power between information rich and information poor.

The availability of computers to small enterprises and individuals in the 1980's had immediate implications for the development of software which became evident following the release of the PDP-8. For the first time computer owners lacked the specialized staffs to develop their own software,

---

[13] Sherry Turkle, The Second Self: Computers & The Human Spirit, (NY, Simon & Schuster, 1984).

and they were numerous so they represented a potentially lucrative market. A new kind of software market, the market of software as a consumer products evolved, and found that customers were anxious to buy. The level of investment in software by users grew rapidly, creating pressure on the manufacturers to maintain stable operating environments, or downwardly compatible ones at least. While this stability benefited their software competitors, the manufacturers accepted the inevitable challenge to their software monopolies, and eventually came to enjoy the benefits of having buyers attracted to their products because of the range of third party software which was available. The transformation of software into a consumer product is another concern of the software archive. Documenting this phenomenon will require not so much the acquisition of these off-the-shelf products, as the documentation of the emerging system for their delivery and the secondary business in consumer accessible on-line data services which they are making possible.

How is it that these new products readily found customers? In effect they used existing outlets. They were reviewed in existing journals and taught in existing schools or through existing user groups. They were touted (and panned) on existing electronic bulletin boards and methods for exploiting them were published by general distribution commercial publishing houses and sold in general book stores. In the process, some of these conduits were transformed as well. One national chain of discount book stores opened separate outlets devoted to software inventories. A recent issue of Science, carried in its new product announcement section, announcements of seven new products, all of which were software based![14]

Of course, not all marketplaces were effected by the emergence of the software industry. Only one customer continued to exist for national security systems and the Federal government's program administration support systems and a small number of customers with highly specialized needs dominated such major arenas as the stock markets or commodity exchanges. In these areas, custom developed software, at increasing expense, continued to be the norm. Often it is still the case that these applications require new hardware to be designed especially for them; thus the market continues to

---

[14] Science, December 19, 1986

witness a stream of special purpose machines ranging from mini-computers to super computers with custom programming. The pressure to reduce the costs of these software applications led to the construction of numerous specialized tools for all aspects of the software development, testing and implementaion process. These tools were then used in the construction of custom solutions and in the development of commercially available software. As a consequence, the market for software workbench products, productivity tools and testing tools itself became an arena for commercial product development. Increasingly powerful tool boxes incorporating higher and higher level features including artificial intelligence applications for self checking began to be produced in commercial quantities. These same large systems also required substantial communication networks which were themselves under the control of sophisticated software systems. For these systems, software development is a vast organized effort akin to major construction projects. There are increasingly specialized roles assigned to participants and growing degrees of similarity in the tasks of designers, project managers and technicians in software projects and other large scale engineering efforts. The software archive will need to continue to document changing methods and tools in software development. As a practical matter, this may involve indentifying software efforts during thier active life and working closely with those responsible for them to assure that adequate documentation is maintained.

As noted earlier, software functions are increasingly being migrated into chips, freeing the memory of machines for more complex software driven functions and decreasing the access or execution time by precious nanoseconds. While the proliferation of software over the past fifteen years presents a challenge to those who would document it, an equally complex challenge derives from the symbiotic relationship between hardware and software. From the earliest computers forward, architects of computer systems embodied in the hardware functions which would otherwise require software instructions. As computers evolved in complexity, the governance of storage heirarchies and input/output devices required increasingly complex operating system. Over time, some specialized applications could be better performed with machines designed to satisfy applications requirements at the hardware level without the overhead of those general operating systems software functions not used by the specialized application. With the

implementation of solid state devices (printed circuit boards and silicon chips) in the place of transistors, which had themselves replaced the vacuum tubes of the earliest computers, a means was found of providing specialized instructions in hardware which had earlier been designed in software. The software archive must be alert to the source of hardware innovations in software and of software innovations in hardware, documenting special purpose computers as the source for software innovations resident on general purpose machines and visa versa.

Analysis of the history of software will always play a significant role in defining the potential scope of a software archive. Such research will, therefore, need to be provided for as an integral component of the program. In itself, such research does not define what any given archive should collect, nor how it should use the materials or who it should serve. The role of historical understanding here is to provide one set of criteria by which software might be collected. A totally different set of criteria are suggested by the view of software taken by an application based observer.

## B. APPLICATIONS AS A FRAMEWORK FOR COLLECTING

While some institutions, such as a computer museum or computing department within a history museum will see collecting the documentation of software history in itself as falling within their mission, most cultural repositories should be considering collecting software archives not to document the history of software per se but to document those domains they already consider as falling within their collecting responsibility. In the terminology used by the computer industry, these institutions would be documenting automated "applications". An application is a software program which supports a specific activity - designing an aircraft, composing music, registering students for classes, or determining eligibility of applicants for insurance reimbursements. In collecting applications software, the focus of the collecting institution is on the role the software played in the activity supported by it, rather than in its contribution to fundamental concepts in the evolution of software or to the economic and social history of software in itself. Since most spheres of human activity since 1970 have been impacted by software applications, most cultural repositories which serve in any way to document human cultural achievements have some reason to consider applications software within their collecting scope.

This view of application software as a documentary record is not the same as, and indeed conflicts in some ways with, the interest among archivists in collecting "machine-readable data files". Since the early 1970's, archival programs in large governmental entities (nations, states, municipalities) professional organizations have been formed, including the Association of Public Data Users (APDU) and the International Association for Social Science Information Service and Technology (I-ASSIST). The same organizational interests and professional concerns have informed the Society of American Archivists Task Force of Automated Records and Techniques and led to the development within the Society of American Archivists of training courses and technical programs devoted to the administration of machine readable records. But these programs are not only not constructing software archives, they have, until very recently, adopted approaches which are hostile to documenting applications software for reasons associated with their need to identify formats for retaining social science data for future use. Because the reasons that data archives have been software collections are relevant to software collecting, and are discussed in more detail later in this report, they are only introduced here in order to explain an otherwise inexplicable attitude.

Machine-readable data archives have consisted of largely of the results of large scale studies (ranging from census data to satellite remote sensing data) or the contents of huge case filing systems (veterans records, accident reports or academic matriculation files). The purpose in collecting these data is to allow future generations of researchers to analyze their information content. In order to preserve the information for future use, the data archivist has, in the past, advised that it be taken out of its software implementation and recorded it in a documented, software independent, flat file tape format. In this way the information can be read back into any system the researcher is using at any time in the future and the only obsolesence concern which must be managed is to assure that the medium (tape for now) can be read by the users. Since magnetic tapes need to be copied every several years in order to preserve the information, this concern can also be addressed in the course of routine maintenance simply by copying tapes over onto newer generations of media.

The approach to saving information which has been adopted by data archives does not retain software in which the information was generated.

Nor does it document the policies and procedures which produced the information, an evidential concern of traditional archivists. When we consider that applications software embodies the instructions which are intended to produce a specific result (e.g. return an 8% interest compounded weekly on investments over $1200), the importance of retaining software as evidence that the advertised or intended consequences are, in fact, being realized, can be appreciated. In governmental settings, or with respect to regulated functions or activities for which a corporate entity might be liable, there are strong legal reasons to retain application software.

Oddly, non-financial applications software is rarely audited on a regular basis, to assure that it is, or ever was, executing policies as intended. Because software is usually being modified on an on-going basis, and because modifications may have unintended consequences, it is not sufficient to test software upon acceptance, even if it was possible to create a test data set which executed every possible part of a application (and this problem, which is an important one for software developers, has yet to be satisfactorily resolved)[15]. Output, in the form of data files which might be of interest to social science researchers and machine-readable data archives, rarely provide any indication of what was actually being done by the software program which generated it.

Finally, there are applications programs which justify retention as information in themselves. Already discussed are those which would be retained because they contribute to our understanding of the history of software (invoking new ideas in data manipulation or strategies for processing) or are creative products equal other output of literature or the arts (including musical compositions and visual presentations). Not yet introduced are software solutions which come to define a business strategy or make possible a social program. Business historians have not yet written extensively on the role which software is playing in strategic positioning of industries and within industries, but it is evident that software plays a major role in competition in a variety of contemporary business (banking,

---

[15] These problems are so serious that an entire sub-discipline in software engineering has developed to assure quality control of software and to maintain a record of all the software changes and their implications. For further information the reader should look at the literature on software configuration management.

airlines, insurance to name only the most obvious) and that the successes and failures of specific firms are directly attributable to their software.[16] Of course, we cannot ignore that software is itself a commodity, and increasingly a consumer product, which might be documented for the same reasons that other products are. Lastly, a number of public policy discussions are a direct consequence of the capabilities of software; for example, the potential for linkages between databases has created significant concerns for personal privacy (some resulting in legislation) and for national security (leading to the recent articulation of the "mosaic" theory whereby appropriately equipped software access to unclassified information sources is held to create sensitive information which might be classifiable). Such software capabilities help to define the environment in which software is received by the culture, and are, therefore, part of the background against which all applications must be understood.

---

[16] The most recent issue of PC Week, August 4, 1987, contains two lead stories which confirm this impression. "LAN Databases Help Bolster Gas-Marketing Efforts at ARCO" leads off "In the highly competitive, post-deregulation natural-gas market, Arco Oil and Gas Co. needed an edge", while the article headlined "Delta Takes Off with Revamped LAN-Based Reservation System" begins "Delta Airlines has developed a PC LAN-based computer-reservation system (CRS) that it hopes will make it the leading information-systems supplier to the nation's 30,000 independent travel agencies."