VI. PLANNING FRAMEWORK

A. A MODEL

How then, should a cultural history repository, go about establishing a software archive?

First, the decision to form a software archive should derive directly from the goals and purposes of the host institution and the criteria for acquisition should fit into the collection development policies of the host. Either the institution must have the commitment to documenting modern cultural history, and a concommitant ability to supporting research (because a software collection will contribute less to exhibition and public programs than many other collections), or it must have the obligation to preserve an evidential record which is embodied in software.

If the archive is to "collect", rather than schedule, software, it will need to define the criteria it will use to evaluate such gifts. In effect, these criteria are identical to those it would employ in any other self-conscious collecting effort, except that they will be extremely difficult to communicate to potential donors who will tend to equate the holdings of a software archive with programming code. Defining such a collecting effort involves identifying either products or persons (including corporations) which exemplify a "type" being sought. Approaching potential donors with a clear definition of the place which the software they possess occupies in the documentation strategy of the archive will also be an incentive to them to donate the desired materials.

Defining the significant developments and actors in any given plane of the matrix of perspectives on the history of software described in Section IIA, is a substantial research effort. The staff of a software archive might use specific exhibitions, funded research projects, special publications or probes of areas of corporate activity, to frame software collecting objectives in specific arenas. The best model for employing such special projects in the achievement of an overall documentation strategy, is the very successful Center for the History of Physics at the American Institute of Physics in New York.[30] The Center, founded in 1965 at the end of a four year survey pro-

---

[30]. Joan Warnow-Blewett, "Documentation Strategy Process: A Case Study", American Archivist, vol 50 #1, Winter 1987, p.29-47

ject funded by NSF, both collects materials relating to the history of physics and encourages other organizations to do so. It assists in identifying materials which should be preserved and maintains a catalog of materials housed in repositories throughout the world. Every several years the Center has launched a study of some special arena for physics research - nuclear physics, astrophysics, solid state physics, physics in national laboratories, and laser physics - with support from oustide funding agencies. These studies have served to identify important historical developments, locate existing documentation and provide information about its contents, alert collecting organizations about lacunae in collected documentation, and to support a growing community of scholars, many of who served post-doctoral years as project staff, who further interpret the field. They have also had the important secondary effect of making potential donors aware that they possess materials of interest to historical repositories and of helping the repositories to define criterial for asessing such evidence if it is offered. The full-time archivist and full-time historian on the staff have had their ranks augmented by project oriented staff throughout the life of the Center, and the project oriented staff have been reponsible for numerous publications and at least one major travelling exhibit which have enhanced the visibility of the repository. Historians of science as well as practicing scientist have proved very supportive and the Center has also made some fundamental contributions to archival practice.

The Center has found that the most interesting documentation of a science or technology from an historical point of view will be that which sheds light on process. Such documentation, of false starts, intermediate steps, ideas accidentally gathered during the course of other day to day activity, and personal recollections, is also the most fragile. Usually this kind of documentatation will survive in the context of undisturbed records but will not be retained after several moves, a change in ownership of the company, the retirement or death of the creator, or any conscious filtering by other than informed archivists. Therefore, the time to undertake such an effort is now, rather than later, and even at that, archivists will discover that much of the documentation is already lost.

The most important documentation from an evidentiary point of view

will be the record of software changes and change orders and the initial documentation of requirements and the programming specifications which were developed to satisfy those requirements. In effect there are always two evidentiary questions: did the software faithfully execute the intended procedures? and What information would normally have been available to a prerson using this system in the intended fashion at a given time?

B. SOFTWARE OPERABILITY

The barrier most often cited by archivists and museum curators to establishing a software archive or museum is the assumption that it will be necessary for the purposes of such an archive to preserve the software code itself in a form which makes it possible to run it on the hardware for which it was designed. If this was not such an insuperable stumbling block in itself, planners would soon discover that the same logic requires the preservation in operating order of all the software which ran in concert with the target package.

If it were necessary to retain some or all software in obsolete formats and to run it on the systems for which it was designed, the facilities required for any software archive would have to include a fully equipped, temperature controlled, fire protected, and heavily staffed computing center (of potentially vast size) and substantial expansion space for the variety of drives, CPU's , and I/O devices which were present in the systems environments for which the software was written. In this scenario the acquisition of applications will require acquisition of operating systems and language compilers which are synchronous with the application software release. Operating such systems would require all the usual expertise of a systems software staff, plus knowledge of a continuous stream of releases of each software component, for each manufacturer, if not each machine, in inventory. Obviously, deciding that it is necessary to preserve software in its context of operation in order to meaningfully study it is tantamount to determining that there cannot be comprehensive software archives for the full range of scholarly research topics.

I have concluded, however, that it is not necessary to be able to "run" the software. What, I began by asking, is the primary purpose of the archive? If the answer is public display or visitor education, then the software must run, or be emulated, to be appreciated. Since the answer is to support schol-

arly research or provide evidence of the functioning of an organization whose procedures are executed by software, the question is considerably more complex.

First we should consider just what is at stake. If we determine that meaningful kinds of historical research on software code require the preservation of associated software and hardware systems in working order, it is exceptionally unlikely that much software will ever be preserved, and that fraction which is will be retained, almost by definition, in museums. If, on the other hand, we save software code and documentation for which the enabling software and hardware tools are lacking, we must know what kinds of scholarly research can be conducted about software without running it and the kinds of associated documentary materials which will best support those types of research.

Let me define the issues, so as to make certain that we are not setting up a straw man. The debate is not about the value of programming code. I presume there is no debate over whether research can be conducted on the commercial distibution of software, or its legal protection, or the mechanisms of social recognition among communities of developers, or the impact of particular products or types of products on spheres of business. Surely no one will dispute that each of these kinds of research can be conducted without code, that each will contribute to understanding the history of software, and that, therefore, documentation to support these kinds of research are appropriately collected by a software archive. Many forms of material, from advertising copy to oral history tapes will contribute to this history, and the underlying story will be appreciated by laymen and will be accessible through exhibits, without recourse to the code.

One variety of scholarly research which will not be satisfied without recourse to the code is called "internalist" history of science or intellectual history. It is concerned with the development of concepts and the ways in which they are articulated. In order to know whether a software designer or programmer employed a paticular recursion technique or constructed independently executing modules, we need to examine the code. To understand a departure in AI programming, we may need to see how LISP demons were employed. However, the fact that many of the questions which intellectual historians might want to ask can only be answered from examining code, does not mean that they could be answered by running a program. Indeed,

the foregoing examples are illustrative of questions which would *not* be clarified by having the program operate; they can only be answered by dissection and analysis. One of the kinds of research which the software archive does want to support is precisely this type of intellectual history. It is for this reason that the historically oriented archive will often wish to acquire the code for specific routines, or modules, even though the systems of which they are components are not historically important or not available (the evidentiary archive would, in general, not retain such fragments).

At the same time, I will not dispute that even though the code, software documentation and advertising campaign records may be of value in understanding why one product succeeded where another failed, we will better appreciate consumer resistance if we could "feel" the product and see the way it worked. Nevertheless, I believe that published sources from the time and the correspondence of software company executives with their clients will point to the source of the market failure, even if we are deprived of a first hand "feel".

What, then, is the utility of saving the software (and the hardware on which it runs) in operating condition? What kinds of research can only be conducted, or can be conducted significantly easier, this way?

It is axiomatic among systems designers that one cannot fully understand a complex system by the study of its parts. Software is, of course, only one component in a complex system consisting of other software, firmware, hardware, communications media, standards, data, people etc. The kinds of interrelationships and dependencies which exist in complex systems are extremely difficult, if not impossible, to comprehend in the abstract. This is, of course, one of the reasons why software undergoes so many releases; capabilities which previously could not be realized become possible because of a change in the software or hardware in which the application is implemented. Changes to the application suggest needs to optimize features of the underlying environment which is then changed in response. As in any ecological system, these adjustments are taking place all the time.

The ecological metaphor suggests that researchers wishing to understand how systems actually worked would be seriously handicapped if the software could not be made to "run" as designed. But even retaining every release of an application system, and all the software and hardware releases which took place over its life, would not allow the researcher to model how

something actually worked at a specific site and time. Systems software specialists are notoriously clever at applying fixes to their local systems and equally poor at documenting them. The way to software actually ran at a particular place and time is nearly impossible to define, to say nothing of trying to replicate it. Even if we assumed that the "configuration management" history of a site was perfectly documented, it is extremely unlikely that we could ever reconstruct it, at least not without a large staff of systems programmers, and even then what we could conclude would be meaningful only for that site.

This is not to suggest that there is no value in seeing how a piece of software ran, or was intended to run. Much can be learned from such an experience, whether as a simulation, on film or by running it on the actual device for which it was designed. It is to argue however that too much can be made of the value of such "live" study of software for research purposes, and that with very few exceptions the costs of achieving such an end will be found to greatly outweigh the benefits.

If this conclusion is sound, there is little reason to retain software code in the original storage media or formats. If we don't have working seven inch magnetic drives attached ot the appropriate system with all the required software and output devices of the period, then we might as well take the seven inch tape and transfer its contents (while this is still possible using commercial service bureaus) to a medium which will be accessible to our research clientele. While code which is in print form need not be made machine readable, code which is already in a machine readable format should almost certainly be maintained in machine readable form on a contemporary medium, if possible (with the current preference being secured directories on hard drives or WORM drives). When the program is copied onto new media, and the original medium is an artifact of intrinsic historical interest, such as the Teletype Tape which input Bill Gates' BASIC interpreter for the Altair computer, the artifact can be accessioned into the Museum. In such a case both the copied code (used for historical research purposes) and the original artifact (used for exhibit purposes) would be kept. A similar case would be made for the first program commercially distributed on five and one-quarter inch floppies or some like distinction, but in most cases the original format is irrelevant.

Obversely, if the computers which ran a particular program are not in

existence anywhere, then retaining the system code would be futile if we believed that software had to be run to be studied. Those with whom I have discussed this issue seem to be in agreement that the argument for keeping code is actually stronger for such extinct hardware environments than it is for hardware environments which have been preserved or documented. While nearly everyone concurred that it would be acceptable for research purposes to retain only those modules, routines or even sub-routines which represented interesting departures with large scale systems written on computers which still exist or have been well documented, they saw benefits to retaining the complete code in the case of systems which don't exist and/or are poorly documented. The complete code will suggest the functions which lie beyond it; if these are running elsewhere, it is not worth keeping another copy, but if they are not documented, the code of an application which calls them may suggest aspects of their design.

C. THE LAW

Anyone considering creating a software archive or museum confronts a number of questions which can best be answered by an appreciation of the legal status of software as a protected asset. Can software owners and developers get protection for their products if they are deposited in an archive organized for historical research? What kinds of uses can the archive permit users to make of the software deposited with them? And who owns the software, and is therefore able to give it to the archive?

The same review of the legal environment surrounding software protection allows us to answer the question of whether copyright and patent office records will be an important source for identifying the universe of software and establishing what novel features of that universe should be considered for archival collection.

The authoritative source for any review of the legal issues surrounding software is The Computer Law Monitor (CLM). CLM reports on Federal court and state superior and supreme court rulings on matters of copyright, patent, trade secret and trademark protection for computer software and firmware (chips, microde etc.) as well as about many other legal matters relating to computing. Since laws, and court interpretations of them, can change, CLM should be considered a continuing source for answering the questions posed above.

As of mid-1987, certain conclusions are warranted. First, while copyright protection has been extended to all software programs in recent years (Apple v. Formula International), the situation has been sufficiently confused until 1985 to limit the utility of copyright office files for a comprehensive analysis of the software universe.

Secondly, the courts have recently upheld strict interpretations of the rights of copyright holders. They have ruled that software is tangible personal property (National Surety Corp. v. Allied Systems) and awarded significant damages for its misappropriation. They have made it clear that printing the code does not reduce rights to protect against its copying in electronic form (Micro-Sparc Inc. v. Amtype Corp., Apple v. Formula International), and they have ruled that reproducing the concepts and flow of code in another language (Whelan Associated v. Jaslow Dental Labs) or even making code from copyrighted English language statements of methods (Williams v. Arndt) is a violation of the act. They have further ruled that the programs need not have a copyright notice on them, if they are distributed with written material bearing the notice (Koontz v. Jaffarian). Finally, thay have applied copyright protection to video games (Midway Mfg. v. Dirkschneider) and other computer instructions so long as these are not determined to be the only way to produce a particular result. These protections should be adequate to assure the future protection of owners of software who deposit such materials as code and the records of its development in archives.

Thirdly, the courts in copyright and trade secret suits have provided positive incentives for archiving the records of the design and development process by ruling that in the absence of such records of independent development, charges that software was copied illegally can be upheld (Dickerman Associates v. Tiverton Bottled Gas). In other decisions about trade secrets, the courts have placed a substantial burden on the holders of trade secrets to demonstrate that they possessed a secret which gave them competitive advantage, that they took measures to prevent the disclosure, that they had a confident relationship with the party charged with disclosure and that when adopted for use the trade secret worked to their financial detriment. In the main, these requirements, along with growing protection in copyright and patents, may have reduced the appeal of trade secrets as a method of protecting software, but they also make it clear that where this method has

been employed, depositing the information in an archive, even with strict instructions to keep it closed until some future date, would risk the protection afforded by secrecy. In these cases it may be necessary for the archive to arrange for a future deposit, with the materials remaining in the custody of the owner, or a third party acting as the owners confident agent, until that date.

Fourthly, patent protection has been extended to ROM chips (Diamond v. Bradley) and may be extended to other forms of microcode (NEC Corp v. Intel Corp. still underway) despite earlier rulings that software, as a "calculation, mathematical formula or algorithm" was not subject to patent. Texas Instruments settled out of court with Fujitsu and Sharp early in January 1987 under terms which suggest that patent protections for RAM chips are now perceived to have real teeth. Now that computer code embodied in firmware is recognized as a "mechanism" and may be patented, we can expect the patent office files to become an increasingly good source for assessment of the direction of firmware development (especially since patentees must argue why their inventions are novel). Patent protection also depends on demonstrated development documentation, so the opening of patent protection also bodes well for archives.

However, the courts have not made the task of archives easier by their rulings on software ownership. In the case of Jostens Inc. v. National Computer Systems Inc. for example, they ruled that the purchaser of a proprietary software package owned the rights to the package as a whole, not to "the individual routines or lines of code", which, by default, belonged to the developer. In S&H Computer Systems v. SAS Institute, the court ruled that SAS was prohibited from copyrighting its software due to having accepted government funding early in the development process (before incorporating) which required products to be in the public domain. In numerous rulings on employment of technical staff, the courts have ruled that software products belong to employers, but that software concepts belong to their inventors. Since an historical archive will be interested both in collecting software products (a relatively straightforward ownership issue) and software concepts (a much more complex issue, since they must be embodied in code which may belong to someone other than the creator), they will need to be aware of these issues and seek the protection of joint donations when that is appropriate. The ownership of software developed under contract or by

employees of a firm may be clear, but the ownership of the copyright is not clear cut at all unless it was specifically given over to the firm which purchased the software or software license.

The upshot is that acquiring title to software may be a stumbling block for a museum or archival repository which wants to establish such a program. The developer, his or her employer and the client all have claims to ownership and each is also likely to possess different parts of the documentation. While such ownership rights are not legally any less straightforward than they are in any other kind of contracted creative production, these kinds of relationships, which have been exceptions in traditional archives, are likely to be very common in software collections. As a practical matter, this may make very little difference so long as the software being collected is obsolete. In such a case, employer and/or client rights are unlikely to be claimed when the source of the materials collected is the creator. Obversely, the creator would probably not retain rights in materials turned over to his or her employer or client, and would be unlikely to claim them against the archive in any case. However, anyone using the archive with the intention of publishing, would be well advised in any copyright situation, to seek all plausible permissions before reusing the material.

Even though the ownership issues are problematic it is not always, or even perhaps generally, advisable to wait until the commercial value of a software product has been passed to collect documentation. Much better records will be collected if the product is identified for acquisition early in its life. Indeed, it is likely to be impossible to find documentation of software once it has ceased being actively used[31]. In order to collect software related materials early in the product life without risk of liability the software archive will probably need to resort to some or all of the following tactics:

* acquire permissions from all parties which can be identified and located.

---

31 The author and Helen Samuels, archivist of M.I.T., recently visited the developers of the first time-sharing system, which was developed at MIT and used there for a decade. These individuals doubted that any documentation of that system had survived, and they had not substantially changed their methods even though they were now directing a project which has the self-conscious goal of defining the next generation of educational software.

* arrange for the acquisition of the materials at a future data when the party which has commercial interest decides that interest has elapsed

* acquire materials as they are being created but keep them closed to research for a period of time adequate to assure that the commercial interest has passed.

* make materials available, whenever they are opened, with strict warnings about ownership rigths in conjunction with initial registration of patrons, each specific request for materials from the collection, and any requests to duplicate portions of the materials.

Of course, the greatest protection derives from being granted ownership and all rights of use by those who previously enjoyed these rights. In acquiring holdings, the software archive should seek donations which grant the most comprehensive rights possible to the archive (but not necessarily to its patrons).[32]

Finally, the courts have been extremely clear about what constitutes fair use of protected (copyright) materials. In this they have clarified or a software archive what it may and may not permit of its users. Fair use includes any access which serves a public purpose such as criticism, comment, news reporting, teaching, scholarship, and research so long as the activity is not for profit and does not harm the potential market value of the work. Since any study of software and its development conducted in an archive would meet these tests, except for a use which resulted in copying the work for resale or reproduction in competing product (covered by the provisions of copyright) it would seem that a software archive is protected in the issue of user access simply by assuring that it is understood that all materials in its custody are covered by copyright - just as in any traditional archive. Exhibit uses, reproduction for explanatory, educational or scholarly quotation, and technical criticism would all be permitted.

---

[32] U.S. Congress, Office of Technology Assessment, Intellectual Property Rights in an Age of Electronics and Information, OTA-CIT-302 (Washington, DC, USGPO, 1986)