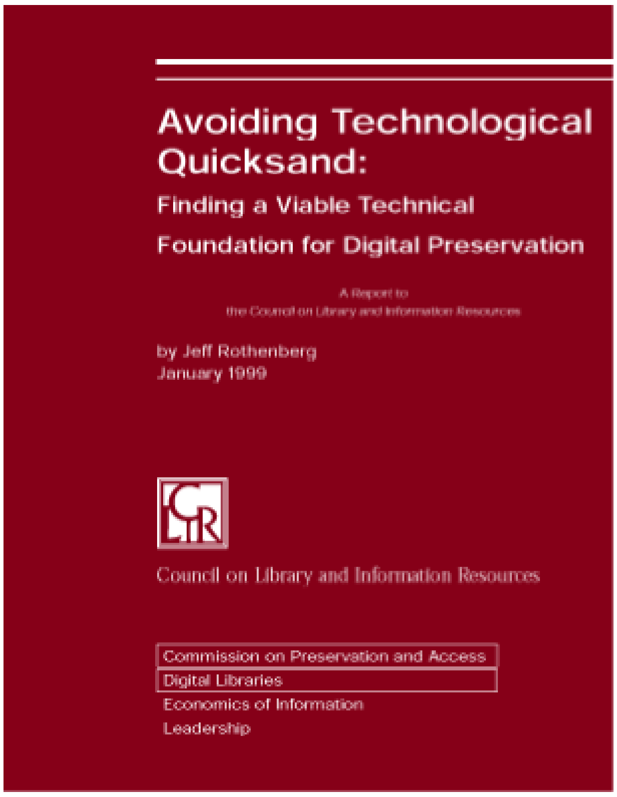

Avoiding Technological Quicksand:

Finding a Viable Technical Foundation for Digital Preservation

A Report to the Council on Library and Information Resources

by Jeff Rothenberg
January 1999

Council on Library and Information Resources

Commission on Preservation and Access
Digital Libraries
Economics of Information
Leadership

Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation

by Jeff Rothenberg
January 1998

Contents

Preface

In 1996, the Commission on Preservation and Access and the Research Libraries Group issued the final report of the Task Force on the Archiving of Digital Information. Chaired by John Garrett and Donald Waters, the task force spent over a year analyzing the problem, considering options, consulting with others around the world, and formulating a series of recommendations. The conclusion reached by the impressive group of 21 experts was alarming—there is, at present, no way to guarantee the preservation of digital information. And it is not simply a technical problem. A serious commitment to preserving digital information requires a legal environment that enables preservation. It also means that specific organizations—libraries, government agencies, corporations—must take responsibility for preservation by enacting new policies and creating the economic means to secure survival of this generation's knowledge into the future.

The Council on Library and Information Resources, which absorbed the Commission on Preservation and Access in July 1997, continues to search for answers to the troubling question of how digital information will be preserved. Raising public awareness is an important goal, and we have pursued it vigorously. Since the publication of the task force report in 1996, we have spoken to library and scholarly groups here and abroad, published a number of papers, and produced an hour-long documentary film on the subject for broadcast on public television. The film especially has made an impression, and several observers have wondered why we have spent so much time in describing the problems and so little in finding solutions.

In fact, we have also been seeking solutions, and the present paper by Jeff Rothenberg is the first in a series resulting from our efforts. Each paper in the series will propose an approach to the preservation of digital information. Each approach addresses the important parts of the problem. We believe that it is best to assemble as many ideas as possible, to place them before a knowledgeable audience, and to stimulate debate about their strengths and weaknesses as solutions to particular preservation problems.

Jeff Rothenberg is a senior research scientist of the RAND Corporation. His paper is an important contribution to our efforts.

Executive Summary

There is as yet no viable long-term strategy to ensure that digital information will be readable in the future. Digital documents are vulnerable to loss via the decay and obsolescence of the media on which they are stored, and they become inaccessible and unreadable when the software needed to interpret them, or the hardware on which that software runs, becomes obsolete and is lost. Preserving digital documents may require substantial new investments, since the scope of this problem extends beyond the traditional library domain, affecting such things as government records, environmental and scientific baseline data, documentation of toxic waste disposal, medical records, corporate data, and electronic-commerce transactions.

This report explores the technical depth of the problem of long-term digital

preservation, analyzes the inadequacies of a number of ideas that have been proposed as solutions, and elaborates the emulation strategy. The central idea of the emulation strategy is to emulate obsolete systems on future, unknown systems, so that a digital document's original software can be run in the future despite being obsolete. Though it requires further research and proof of feasibility, this approach appears to have many advantages over the other approaches suggested and is offered as a promising candidate for a solution to the problem of preserving digital material far into the future. Since this approach was first outlined, it has received considerable attention and, in the author's view, is the only approach yet suggested to offer a true solution to the problem of digital preservation.

The long-term digital preservation problem calls for a long-lived solution that does not require continual heroic effort or repeated invention of new approaches every time formats, software or hardware paradigms, document types, or recordkeeping practices change. The approach must be extensible, since we cannot predict future changes, and it must not require labor-intensive translation or examination of individual documents. It must handle current and future documents of unknown type in a uniform way, while being capable of evolving as necessary. Furthermore, it should allow flexible choices and tradeoffs among priorities such as access, fidelity, and ease of document management.

Most approaches that have been suggested as solutions—printing digital documents on paper, relying on standards to keep them readable, reading them by running obsolete software and hardware preserved in museums, or translating them so that they "migrate" into forms accessible by future generations of software—are labor-intensive and ultimately incapable of preserving digital documents in their original forms.

The best way to satisfy the criteria for a solution is to run the original software under emulation on future computers. This is the only reliable way to recreate a digital document's original functionality, look, and feel. Though it may not be feasible to preserve every conceivable attribute of a digital document in this way, it should be possible to recreate the document's behavior as accurately as desired—and to test this accuracy in advance.

The implementation of this emulation approach involves: (1) developing generalizable techniques for specifying emulators that will run on unknown future computers and that capture all of those attributes required to recreate the behavior of current and future digital documents; (2) developing techniques for saving—in human-readable form—the metadata needed to find, access, and recreate digital documents, so that emulation techniques can be used for preservation; and (3) developing techniques for encapsulating documents, their attendant metadata, software, and emulator specifications in ways that ensure their cohesion and prevent their corruption. The only assumption that this approach makes about future computers is that they will be able to perform any computable function and (optionally) that they will be faster and/or cheaper to use than current computers.

Next Previous

Return to CLIR Home Page >>

1. Introduction

In a very short time, preservation has developed into a critically important part of managing a library's most precious assets, its collections.
—Abby Smith, "Preservation in the Future Tense"

The vision of creating digital libraries that will be able to preserve our heritage currently rests on technological quicksand. There is as yet no viable long-term strategy to ensure that digital information will be readable in the future. Not only are digital documents vulnerable to loss via media decay and obsolescence, but they become equally inaccessible and unreadable if the software needed to interpret them—or the hardware on which that software runs—is lost or becomes obsolete.

This report explores the technical depth of this problem, analyzes the inadequacies of a number of ideas that have been proposed as solutions, and elaborates the emulation strategy, which is, in my view, the only approach yet suggested to offer a true solution to the problem of digital preservation (Rothenberg 1995a). Other proposed solutions involve printing digital documents on paper, relying on standards to keep them readable, reading them by running obsolete software and hardware preserved in museums, or translating them so that they "migrate" into forms accessible by future generations of software. Yet all of these approaches are short-sighted, labor-intensive, and ultimately incapable of preserving digital documents in their original forms. Emulation, on the other hand, promises predictable, cost-effective preservation of original documents, by means of running their original software under emulation on future computers.

2. The Digital Longevity Problem

Documents, data, records, and informational and cultural artifacts of all kinds are rapidly being converted to digital form, if they were not created digitally to begin with. This rush to digitize is being driven by powerful incentives, including the ability to make perfect copies of digital artifacts, to publish them on a wide range of media, to distribute and disseminate them over networks, to reformat and convert them into alternate forms, to locate them, search their contents, and retrieve them, and to process them with automated and semi-automated tools. Yet the longevity of digital content is problematic for a number of complex and interrelated reasons (UNACCIS 1990, Lesk 1995, Morris 1998, Popkin and Cushman 1993, Rothenberg 1995b, Getty 1998).

It is now generally recognized that the physical lifetimes of digital storage media are often surprisingly short, requiring information to be "refreshed" by copying it onto new media with disturbing frequency. The technological obsolescence of these media (and of the hardware and software necessary to read them) poses a different and equally urgent threat. Moreover, most digital documents and artifacts exist only in encoded form, requiring specific software to bring their bit streams to life and make them truly usable; as these programs (or the hardware/software environments in which they run) become obsolete, the digital documents that depend on them become unreadable—held hostage to their own encoding. This problem is paradoxical, given the fact that digital documents can be copied perfectly, which is often naively taken to mean that they are eternal. This paradox prompted my ironic observation (Rothenberg 1997), "Digital documents last forever—or five years, whichever comes first." There is currently no demonstrably viable technical solution to this problem; yet if it is not solved, our increasingly digital heritage is in grave risk of being lost (Michelson and Rothenberg 1992, Morelli 1998, Swade 1998).

In addition to the technical aspects of this problem, there are administrative, procedural, organizational, and policy issues surrounding the management of digital material. Digital documents are different from traditional paper documents in ways that have significant implications for the means by which they are generated, captured, transmitted, stored, maintained, accessed, and managed. Paramount among these differences is the greatly reduced lifetime of digital information without some form of active preservation. This mandates new approaches to accessioning and saving digital documents to avoid their loss. These approaches raise nontechnical issues concerning jurisdiction, funding, responsibility for successive phases of the digital document life cycle, and the development of policies requiring adherence to standard techniques and practices to prevent the loss of digital information. However, few of these nontechnical issues can be meaningfully addressed in the absence of a sound, accepted technical solution to the digital longevity problem.

3. Preservation in the Digital Age

> The goal of any preservation program is to ensure long-term, ready access to the information resources of an institution.
> —Abby Smith, "Preservation in the Future Tense"

Preservation constitutes one leg of a tripod that supports informational institutions such as libraries, the other legs being access and the development and management of collections (Fox and Marchionini 1998, Schurer 1998). Without preservation, access becomes impossible, and collections decay and disintegrate.

Informational artifacts include documents, data, and records of all kinds, in all media, which I refer to as "documents" here, for simplicity (Roberts 1994). The essence of preserving informational artifacts is the retention of their meaning. This requires the ability to recreate the original form and function of a document when it is accessed, for example, to establish its authenticity, validity, and evidential value and to allow the document's users to understand how its creator and original viewers saw it, what they were (and were not) able to infer from it, what insights it may have conveyed to them, and what aesthetic value it may have had for them.

My focus is on digital documents, by which I mean informational artifacts, some or all aspects of whose intended behavior or use rely on their being encoded in digital form. The term "digital" in this context denotes any means of representing sequences of discrete symbolic values—each value having two or more unambiguously distinguishable states—so that they can, at least in principle, be accessed, manipulated, copied, stored, and transmitted entirely by mechanical means, with high reliability.[1]

The preservation of digital documents involves a number of distinctive requirements. In particular, all such documents possess a unique collection of core digital attributes that must be retained. These attributes include their ability to be copied perfectly, to be accessed without geographic constraint, to be disseminated at virtually no incremental cost (given the existence of appropriate digital infrastructure), and to be machine-readable so that they can be accessed, searched, and processed by automated mechanisms that can modify them, reformat them, and perform arbitrary computations on their contents in all phases of their creation and distribution. Furthermore, new inherently digital ("born-digital") document forms, such as dynamic, distributed, interactive hypertext and hypermedia, must retain their unique functionality, including their ability to integrate information from disparate traditional sources, such as books, periodicals, newspapers, mail, phone messages, data, imagery, and video (Bearman 1991, Bearman 1992, Bikson 1997, Kenney 1997, Michelson and Rothenberg 1992).

In response to the difficulty of saving digital documents (due to factors such as media decay and software and hardware obsolescence, discussed in detail below) it is sometimes suggested that they be printed and saved in hard-copy form. This is a rear-guard action and not a true solution. Printing any but the simplest, traditional documents results in the loss of their unique functionality (such as dynamic interaction, nonlinearity, and integration), and printing any document makes it no longer truly machine-readable, which in turn destroys its core digital attributes (perfect copying, access, distribution, and so forth). Beyond this loss of functionality, printing digital documents sacrifices their original form, which may be of unique historical, contextual, or evidential interest (Bearman 1993, Hedstrom 1991, U. S. District Court for the District of Columbia 1993).

Proposed alternatives to printing digital documents include translating digital documents into standard forms or extracting their contents without regard to their original nature. Though these approaches have traditional analogues (such as the translation of ancient texts into the vernacular to give them a larger audience), they are fraught with danger. The meaning of a document may be quite fragile, since meaning is in the eye of the beholder: what may be a trivial transformation to a casual reader may be a disastrous loss to a scholar, historian, or lawyer. Examples of loss of meaning abound in our daily experience of converting digital documents from their native form into that of some other software application in order to read them. At its best, such conversion often sacrifices subtleties (such as format, font, footnotes, cross-references, citations, headings, numbering, shape, and color); at its worst, it leaves out entire segments (such as graphics, imagery, and sound) or produces meaningless garbage (Horsman 1994).

While it is often useful to create contemporary vernacular transcriptions of historical documents (such as Shakespeare's Sonnets or the Declaration of Independence), society places a high value on retaining the originals so that we may verify that content has not been lost in transcription (whether inadvertently or for nefarious ends), as well as for scholarly and aesthetic purposes. For digital documents, retaining an original may not mean retaining the original medium (which rapidly decays and becomes obsolete), but it should mean retaining the functionality, look, and feel of the original document.

The skills and judgment developed in preservation professionals—the ability to discover the original form of an object and the intent of its creator, and to prolong the life of the object or return the object as nearly as possible to its state at the time of its creation—are precisely the same skill sets that are needed for the future, albeit practiced in a radically different context.
—Abby Smith, "Preservation in the Future Tense"

References

[1] I use the term *digital information* in preference to *electronic information* because it more accurately captures the essential aspects of the problem. Digital information can in principle be represented in nonelectronic form, for example, by using optical or quantum techniques, whereas electronic information is not necessarily digital.

Next Previous

Return to CLIR Home Page >>

4. The Scope of the Problem

The preservation of digital documents is a matter of more than purely academic concern. A 1990 House of Representatives report cited a number of cases of significant digital records that had already been lost or were in serious jeopardy of being lost (U. S. Congress 1990), and the 1997 documentary film *Into the Future* (Sanders 1997) cited additional cases (Bikson 1994, Bikson and Law 1993, Fonseca, Polles and Almeida 1996, Manes 1998, NRC 1995).

In its short history, computer science has become inured to the fact that every new generation of software and hardware technology entails the loss of information, as documents are translated between incompatible formats (Lesk 1992). The most serious losses are caused by paradigm shifts (such as those between networked, hierarchical, and relational databases), which often require the complete redesign of documents (or databases) to migrate to the new paradigm. Whenever this happens, documents that are not in continuing use may well be orphaned by the demise of their original formats, that is, they may be abandoned to save the cost of migration, while each document that does migrate may be turned into something unrecognizably different from its original, which is generally lost.[2] Even when migration does not involve a major paradigm shift, it can result in subtle losses of context and content. Yet aside from this ad hoc migration process, there exists no proven strategy for preserving (or translating) documents over time (Bikson and Frinking 1993).

The scope of this problem extends beyond the traditional library domain, affecting government records, environmental and scientific baseline data, documentation of toxic waste disposal, medical records (whose lifetime must exceed 100 years), corporate data (such as the documentation of drug trials for pharmaceutical companies or of geologic survey data for petrochemical companies), electronic-commerce transactions, and electronic records needed to support forthcoming digital government initiatives (Bikson 1994, Erlandsson 1994).

The library and archives communities have identified at least some critical aspects of this problem and have recognized that preserving digital documents may require substantial new investments and commitments by institutions and government agencies (Bikson forthcoming, Hedstrom 1993, Indiana University 1997, Waters and Garrett 1996). Showing admirable foresight, these communities have begun to discuss alternative economic and administrative policies for funding and managing digital preservation and have begun to develop conceptual frameworks for metadata that are not restricted to the print medium (Day 1997, Dublin Core Metadata Initiative 1998, IEEE 1997, Giguere 1996, Rothenberg 1996). Yet the lack of any long-term technical solution to the problem of digital preservation limits the efficacy of such explorations, since attempting to allocate responsibilities and assess costs for a nonexistent process is of questionable value.

4.1 The need for triage

The practical problem of digital preservation can be viewed at three different time scales. In the short term, many organizations are faced with an urgent need to save digital material that is in imminent danger of becoming unreadable or inaccessible, or to retrieve digital records that are already difficult to access. Yet the often heroic efforts needed to save or retrieve such material may not be generally applicable to preserving digital documents far into the future, and the techniques employed may not even be generalizable to solve similar urgent problems that may arise in the future. These short-term efforts therefore do not provide much leverage, in the sense that they are not replicable for different document types, though they may still be necessary for saving crucial records. In the medium term, organizations must quickly implement policies and technical procedures to prevent digital records from becoming vulnerable to imminent loss in the near future. For the vast bulk of records—those being generated now, those that have been generated fairly recently, or those that have been translated into formats and stored on media that are currently in use—the medium-term issue is how to prevent these records from becoming urgent cases of imminent loss within the next few years, as current media, formats, and software evolve and become obsolete.

In the long term (which is the focus of this report), it is necessary to develop a truly long-lived solution to digital longevity that does not require continual heroic effort or repeated invention of new approaches every time formats, software or hardware paradigms, document types, or recordkeeping practices change. Such an approach must be extensible, in recognition of the fact that we cannot predict future changes, and it must not require labor-intensive (and error-prone) translation or examination of individual records. It must handle current and future records of unknown type in a uniform way, while being capable of evolving as necessary.

4.2 Types of digital information affected

Though most early digital documents have consisted largely of text, the generation of multimedia records has increased rapidly in recent years, to include audio recordings, graphical charts, photographic imagery, and video presentations, among others. In the digital domain, all of these media can be combined into hypermedia records, whose impact and expressivity can be expected to stimulate their increased use. Whereas the bulk of existing digital documents may be textual, multimedia and hypermedia records are likely to become ever more popular and may well become dominant in the near future. Any solution to digital preservation that is limited to text will therefore quickly become obsolete. A true long-term solution should be completely neutral to the form and content of the digital material it preserves.

4.3 Contextual issues

The preservation and management of digital records involves interrelated technical, administrative, procedural, organizational, and policy issues, but a sound technical approach must form the foundation on which everything else rests. Preserving digital records may require substantial new investments and commitments by organizations, institutions and agencies, forcing them to adopt new economic and administrative policies for funding and managing digital preservation. Yet it is impossible to allocate responsibilities or assess costs for an undefined process: until a viable technical approach to digital longevity has been identified and developed, it is premature to spend much effort attempting to design the administrative and organizational environment that will embed whatever technical approach is ultimately adopted.

5. Technical Dimensions of the Problem

Digital media are vulnerable to loss by two independent mechanisms: the physical media on which they are stored are subject to physical decay and obsolescence, and the proper interpretation of the documents themselves is inherently dependent on software.

5.1 Digital media suffer from physical decay and obsolescence

There is reasonably widespread (though by no means universal) awareness of the fact that digital storage media have severely limited physical lifetimes. The National Media Lab has published test results for a wide range of tapes, magnetic disks, CD-ROMs, and other media (Van Bogart 1996), showing that a tape, disk, or even CD that is picked at random (that is, without prior evaluation of the vendor or the specific batch of media) is unlikely to have a lifetime of even five years (Lancaster 1986). Vendors and media scientists may argue vehemently about such numbers, but accurate estimates are ultimately largely irrelevant, since the physical lifetime of media is rarely the constraining factor for digital preservation. Even if archival quality media were introduced in the market, they would probably fail, since they would quickly be made obsolete—despite their physical longevity—by newer media having increased capacity, higher speed, greater convenience, and lower price (Schurer 1998). This is a natural outgrowth of the exponential improvement in storage density, speed, and cost that has characterized digital media development for the past several decades: the market makes older storage media obsolete as newer, better media become available. The short lifetimes of eight-inch floppy disks, tape cartridges and reels, hard-sectored disks, and seven-track tapes, among others, demonstrate how quickly storage formats become inaccessible.

Media obsolescence manifests itself in several ways: the medium itself disappears from the market; appropriate drives capable of reading the medium are no longer produced; and media-accessing programs (device drivers) capable of controlling the drives and deciphering the encodings used on the medium are no longer written for new computers. Upgrading to a new computer system therefore often requires abandoning an old storage medium, even if an organization still has documents stored on that medium.

The dual problems of short media lifetime and rapid obsolescence have led to the nearly universal recognition that digital information must be copied to new media (refreshed) on a very short cycle (every few years). Copying is a straightforward solution to these media problems, though it is not trivial: in particular, the copy process must avoid corrupting documents via compression, encryption, or changing data formats.

In addition, as media become more dense, each copy cycle aggregates many disks, tapes, or other storage units onto a single new unit of storage (say, a CD or its successor, a DVD, digital video disk or digital versatile disk). This raises the question of how to retain any labeling information and metadata that may have been associated with the original media: since it is infeasible to squeeze the contents of the labels of 400 floppy disks to fit on the label of a single CD, label information must be digitized to ensure that it continues to accompany the records it describes. But whereas labels are directly human-readable, digitized information is not; labels and metadata must therefore be digitized in such a way that they remain more easily readable by humans than are the documents they describe. This may seem a relatively trivial aspect of the problem, but it has serious implications (Bearman 1992), as discussed below.

5.2 Digital documents are inherently software-dependent

Though media problems are far from trivial, they are but the tip of the iceberg. Far more problematic is the fact that digital documents are in general dependent on application software to make them accessible and meaningful. Copying media correctly at best ensures that the original bit stream of a digital document will be preserved. But a stream of bits cannot be made self-explanatory, any more than hieroglyphics were self-explanatory for the 1,300 years before the discovery of the Rosetta Stone. A bit stream (like any stream of symbols) can represent anything: not just text but also data, imagery, audio, video, animated graphics, and any other form or format, current or future, singly or combined in a hypermedia lattice of pointers whose formats themselves may be arbitrarily complex and idiosyncratic. Without knowing what is intended, it is impossible to decipher such a stream. In certain restricted cases, it may be possible to decode the stream without additional knowledge: for example, if a bit stream is known to represent simple, linear text, deciphering it is amenable to cryptographic techniques. But in general, a bit stream can be made intelligible only by running the software that created it, or some closely related software that understands it.

This point cannot be overstated: in a very real sense, digital documents exist only by virtue of software that understands how to access and display them; they come into existence only by virtue of running this software.

> When all data are recorded as 0s and 1s, there is, essentially, no object that exists outside of the act of retrieval. The demand for access creates the 'object,' that is, the act of retrieval precipitates the temporary reassembling of 0s and 1s into a meaningful sequence that can be decoded by software and hardware.
> —Abby Smith, "Preservation in the Future Tense"

As this statement implies, the only reliable way (and often the only possible way) to access the meaning and functionality of a digital document is to run its original software—either the software that created it or some closely related software that understands it (Swade 1998). Yet such application software becomes obsolete just as fast as do digital storage media and media-accessing software. And although we can save obsolete software (and the operating system environment in which it runs) as just another bit stream, running that software requires specific computer hardware, which itself becomes obsolete just as quickly. It is therefore not obvious how we can use a digital document's original software to view the document in the future on some unknown future computer (which, for example, might use quantum rather than binary states to perform its computations). This is the crux of the technical problem of preserving digital documents.

5.3 Additional considerations

Any technical solution must also be able to cope with issues of corruption of information, privacy, authentication, validation, and preserving intellectual property rights. This last issue is especially complex for documents that are born digital and therefore have no single original instance, since traditional notions of copies are inapplicable to such documents. Finally, any technical solution must be feasible in terms of the societal and institutional responsibilities and the costs required to implement it.

References

[2] Documents whose continued use is crucial to the individuals or organizations that own them are more likely to be included in the migration process, in recognition of their importance, but this does not guarantee that their meaning will not be inadvertently lost or corrupted.

Next Previous

Return to CLIR Home Page >>

6. The Inadequacy of Most Proposed Approaches

Most approaches that have been proposed fall into one of four categories: (1) reliance on hard copy, (2) reliance on standards, (3) reliance on computer museums, or (4) reliance on migration. Though some of these may play a role in an ultimate solution, none of them comes close to providing a solution by itself, nor does their combination.

6.1 Reliance on hard copy

It is sometimes suggested that digital documents be printed and saved as hard copy. This is not a true solution to the problem, since many documents (especially those that are inherently digital, such as hypermedia) cannot meaningfully be printed at all, or would lose many of their uniquely digital attributes and capabilities if they were printed. Even digital renditions of traditional documents (such as linear text) lose their core digital attributes by being printed; that is, they sacrifice direct machine-readability, which means they can no longer be copied perfectly, transmitted digitally, searched or processed by computer programs, and so forth. Similarly, attempting to save digital documents by printing the 0s and 1s of their bit streams on paper (or engraving them in metal) sacrifices their machine-readability and the core digital attributes that it enables (Bearman 1993, U. S. District Court for the District of Columbia 1993).

Moreover, any such scheme destroys whatever interactive or dynamic functionality an inherently digital document (or a digital rendition of a traditional document) may have, since the document's original software can no longer be run. (While a human reader might be able to ignore the errors introduced by the optical scanning and character recognition of a printed version of a digital document, computer programs are far less tolerant of the errors that this process would introduce when scanning a printed sequence of 0s and 1s.) For all of these reasons, saving digital documents by printing them (whether rendering their content or printing their bit streams) does not offer a solution to the true problem of digital preservation.

6.2 Reliance on standards

On the face of it, reliance on standards appears to offer a solution by allowing digital documents to be represented in forms that will endure into the future and for which future software will always provide accessibility. One version of this argument offers the relational database (RDB) as a paradigmatic example of how this might work (NARA 1991, Thibodeau 1991). This argument contends that since all relational database management systems (RDBMSs) are based on the same mathematical foundation (Codd 1982), any RDB that is accessioned by an institution can be translated without loss into the specific RDB form recognized by the RDBMS used by that institution. Even if the institution later changes RDBMSs, all of its RDBs should be able to migrate to the new RDBMS without loss, since all RDBMSs support the same baseline functionality (UNACCIS 1990, 1992).

While this argument appears convincing, it fails in several significant ways. First, precisely because the relational model legislates a standard baseline of functionality, real RDBMSs must distinguish themselves in the marketplace by introducing proprietary features that extend the relational model (such as "outer" joins, support for views, unique diagramming methods for data modeling, and the like). Any RDB that makes use of such proprietary features becomes at least somewhat nonstandard and will lose some of its functionality if translated into some other RDBMS (Bikson and Frinking 1993). In this way standardization sows the seeds of its own destruction by encouraging vendors to implement nonstandard features in order to secure market share. Users are motivated to use such features because they provide enhanced functionality, but using these features produces nonstandard databases that are likely to be orphaned by reliance on standards, since standards enforce strict limitations on what they can represent, and thereby preserve.

In addition, far from being a paradigmatic example, the relational database is actually unique: no other kind of digital document rests on an underlying formal mathematical foundation that can serve as the basis for its standardization. Word-processing documents, spreadsheets, graphics and image files, hypermedia, animation, and audio and video formats are still evolving so rapidly that it is unrealistic to expect definitive standards for any of these forms to emerge in the near future. Though standards continue to be developed for many kinds of digital documents, they are almost always informal, ad hoc, and relatively short-lived. Moreover, since they lack a formal underpinning, they often compete with rival standards produced for different purposes or by different groups. This leads to the sad but true statement, attributed to Andrew S. Tanenbaum, "One of the great things about standards is that there are so many different ones to choose from!"

Finally, the relational database example demonstrates a fundamental flaw in the standards-based approach. Just as the relational paradigm replaced earlier network and hierarchical database paradigms, it is currently under attack by the new object-oriented database (OODB) paradigm, which may well replace it or at least relegate it to the role of a low-level storage mechanism hidden beneath future object-oriented database management systems (OODBMSs). As was the case with previous database paradigm shifts, the transition from relational to object-oriented databases cannot be made simply by automatically translating RDBs into OODBs. The paradigms are so different that such translation is typically meaningless: even when it is possible, the result is likely to possess neither the formal rigor of the original relational form nor the enhanced semantic expressiveness of the new object-oriented form. This illustrates the fact that even the best standards are often bypassed and made irrelevant by the inevitable paradigm shifts that characterize information science—and will continue to do so.

Proponents of standards often argue that the way to deal with the problem of paradigm shifts is to force digital documents into current standard forms (even if this sacrifices some of their functionality) and then translate them, when current standards become obsolete, into whatever standards supplant the obsolete ones.[3] This is analogous to translating Homer into modern English by way of every intervening language that has existed during the past 2,500 years. The fact that scholars do not do this (but instead find the earliest original they can, which they then translate directly into the current vernacular) is indicative of the fact that something is always lost in translation. Rarely is it possible to recover the original by retranslating the translated version back into the original language.

Not even character encodings last forever: ASCII (the venerable 7-bit American Standard Code for Information Interchange) is slowly giving way to Unicode (a newer 16-bit character set). Furthermore, the history of these encodings shows that successful standards do not always subsume their competitors, as exemplified by the ascendance of ASCII over EBCDIC (the 8-bit Extended Binary Coded Decimal Interchange Code long used by IBM) and the APL character set (designed for Iverson's "A Programming Language"), despite the fact that ASCII cannot represent all of their characters.

Nevertheless, standards should not be dismissed. Some standards (notably standard generalized markup language, SGML, and its offspring) have proven highly extensible and worthwhile within their limited scope. Since text is likely always to be a part of most documents, SGML provides a useful capability (Coleman and Willis 1997), even though it does not by itself solve the problems of nontextual representation or of representing dynamic, interactive documents. In fact, if SGML had been adopted as a common interlingua (a commonly translatable intermediate form) among word processing programs, it would have greatly relieved the daily conversion problems that plague most computer users; yet this has not occurred, implying that even well-designed standards do not necessarily sweep the marketplace (Bikson 1997). Nevertheless, converting digital documents into standard forms, and migrating to new standards if necessary, may be a useful interim approach while a true long-term solution is being developed. I also suggest below that standards may play a minor role in a long-term solution by providing a way to keep metadata and annotations readable.

6.3 Reliance on computer museums

To avoid the dual problems of corruption via translation and abandonment at paradigm shifts, some have suggested that computer museums be established, where old machines would run original software to access obsolete documents (Swade 1998). While this approach exudes a certain technological bravado, it is flawed in a number of fundamental ways. It is unlikely that old machines could be kept running indefinitely at any reasonable cost, and even if they were, this would limit true access to the original forms of old digital documents to a very few sites in the world, thereby again sacrificing many of these documents' core digital attributes.

Furthermore, this approach ignores the fact that old digital documents (and the original software needed to access them) will rarely survive on their original digital media. If an obsolete digital document and its software survive into the future, this will probably be because their bit streams have been copied onto new media that did not exist when the document's original computer was current. For example, an old word processing file from a 1970s personal computer system will not still exist on the 8-inch floppy disk that was native to that system but will instead have migrated onto a 3.5 inch floppy, a CD-ROM, or perhaps a DVD. The obsolete document would therefore have to be read by an obsolete machine from a new medium for which that machine has no physical drive, no interface, and no device software. The museum approach would therefore require building unique new device interfaces between every new medium and every obsolete computer in the museum as new storage media evolve, as well as coding driver software for these devices, which would demand maintaining programming skills for each obsolete machine. This seems hopelessly labor-intensive and ultimately infeasible.

Finally, computer chips themselves have limited physical lifetimes. Integrated circuits decay due to processes such as metal migration (the traces that define circuit connections on the chips migrate through the substrate over time) and dopant diffusion (the atoms that make semiconductors semiconduct diffuse away over time). Even if obsolete computers were stored carefully, maintained religiously, and never used, aging processes such as these would eventually render them inoperative; using them routinely to access obsolete digital documents would undoubtedly accelerate their demise.

One role that computer museums might play in preservation is to perform heroic efforts to retrieve digital information from old storage media. If an old disk or tape is found that may indeed still have readable information on it, an obsolete machine in the museum (which would presumably have a drive and software for the medium in question) could be used in a last-ditch attempt to tease the bits off the medium, as an alternative to electron microscopy or other equally extreme measures. A second role for computer museums might be in verifying the behavior of emulators, as discussed below. Beyond these limited roles, however, computer museums do not appear to be a serious option for the long-term preservation of digital documents.

6.4 Reliance on migration

The approach that most institutions are adopting (if only by default) is to expect digital documents to become unreadable or inaccessible as their original software becomes obsolete and to translate them into new forms as needed whenever this occurs (Bikson and Frinking 1993, Dollar 1992). This is the traditional migration approach of computer science. While it may be better than nothing (better than having no strategy at all or denying that there is a problem), it has little to recommend it.[4]

Migration is by no means a new approach: computer scientists, data administrators and data processing personnel have spent decades performing migration of data, documents, records, and programs to keep valuable information alive and usable. Though it has been employed widely (in the absence of any alternative), the nearly universal experience has been that migration is labor-intensive, time-consuming, expensive, error-prone, and fraught with the danger of losing or corrupting information. Migration requires a unique new solution for each new format or paradigm and each type of document that is to be converted into that new form. Since every paradigm shift entails a new set of problems, there is not necessarily much to be learned from previous migration efforts, making each migration cycle just as difficult, expensive, and problematic as the last. Automatic conversion is rarely possible, and whether conversion is performed automatically, semiautomatically, or by hand, it is very likely to result in at least some loss or corruption, as documents are forced to fit into new forms.

As has been proven repeatedly during the short history of computer science, formats, encodings, and software paradigms change often and in surprising ways. Of the many dynamic aspects of information science, document paradigms, computing paradigms, and software paradigms are among the most volatile, and their evolution routinely eludes prediction. Relational and object-oriented databases, spreadsheets, Web-based hypermedia documents, e-mail attachments, and many other paradigms have appeared on the scene with relatively little warning, at least from the point of view of most computer users. Each new paradigm of this kind requires considerable conversion of programs, documents, and work styles, whether performed by users themselves or by programmers, data administrators, or data processing personnel.

Even though some new paradigms subsume the ones they replace, they often still require a significant conversion effort. For example, the spreadsheet paradigm subsumes simple textual tables, but converting an existing table into a meaningful spreadsheet requires defining the formulas that link the entries in the table, although these relationships are likely to have been merely implicit in the original textual form (and long since forgotten). Similarly, word processing subsumes simple text editing, but conversion of a document from a simple textual form into a specific word processing format requires that fonts, paragraph types, indentation, highlighting, and so forth, be specified, in order to make use of the new medium and avoid producing a result that would otherwise be unacceptably old-fashioned, if not illegible.

One of the worst aspects of migration is that it is impossible to predict what it will entail. Since paradigm shifts cannot be predicted, they may necessitate arbitrarily complex conversion for some or all digital documents in a collection. In reality, of course, particularly complex conversions are unlikely to be affordable in all cases, leading to the abandonment of individual documents or entire corpora when conversion would be prohibitively expensive.

In addition, as when refreshing media, there is a degree of urgency involved in migration. If a given document is not converted when a new paradigm first appears, even if the document is saved in its original form (and refreshed by being copied onto new media), the software required to access its now-obsolete form may be lost or become unusable due to the obsolescence of the required hardware, making future conversion difficult or impossible. Though this urgency is driven by the obsolescence of software and hardware, rather than by the physical decay and obsolescence of the media on which digital documents are stored, it is potentially just as crucial. Therefore migration cannot generally be postponed without incurring the risk that it may become impossible in the future, and that the documents may be irretrievably lost. Worse yet, this problem does not occur just once for a given document (when its original form becomes obsolete) but recurs throughout the future, as each form into which the document has migrated becomes obsolete in turn.

Furthermore, because the cycles of migration that must be performed are determined by the emergence of new formats or paradigms, which cannot be controlled or predicted, it is essentially impossible to estimate *when* migration will have to be performed for a given type of document—the only reliable prediction being that any given type of document is very likely to require conversion into some unforeseeable new form within some random (but probably small) number of years. Since different format changes and paradigm shifts affect different (and unpredictable) types of documents, it is likely that some of the documents within a given corpus will require migration before others, unless the corpus consists entirely of a single type of document (which becomes less likely as documents make increasing use of hypermedia, since a single hypermedia document consists of component subdocuments of various types). This implies that any given corpus is likely to require migration on an arbitrarily (and uncontrollably) short cycle, determined by whichever of the component types of any of its documents is the next to be affected by a new format or paradigm shift.

Finally, migration does not scale well. Because it is labor-intensive and highly dependent on the particular characteristics of individual document formats and paradigms, migration will derive little benefit from increased computing power. It is unlikely that general-purpose automated or semiautomated migration techniques will emerge, and if they do, they should be regarded with great suspicion because of their potential for silently corrupting entire corpora of digital documents by performing inadvertently destructive conversions on them. As the volume of our digital holdings increases over time, each migration cycle will face a greater challenge than the last, making the essentially manual methods available for performing migration increasingly inadequate to the task.

In summary, migration is essentially an approach based on wishful thinking. Because we cannot know how things will change in the future, we cannot predict what we will have to do to keep a given digital document (or type of document) accessible and readable. We can merely predict that document formats, software and hardware will become obsolete, that paradigms will shift in unpredictable ways (unpredictability being the essence of a paradigm shift), and that we will often have to do something. Since different changes may apply to different kinds of documents and records, we must expect such changes to force the migration of different kinds of documents on independent, unrelated schedules, with each document type (and perhaps even individual documents or their components) requiring specialized, labor-intensive handling. We cannot predict how much effort, time, or expense migration will require, how successful it will be in each case, how much will be lost in each conversion, nor how many of our documents will be corrupted, orphaned, or lost in each migration cycle. Furthermore, we can expect each cycle to be a unique experience that derives little or no benefit or cost savings from previous cycles, since each migration will pose a new set of unique problems.

In the absence of any alternative, a migration strategy may be better than no strategy at all; however, to the extent that it provides merely the illusion of a solution, it may in some cases actually be worse than nothing. In the long run, migration promises to be expensive, unscalable, error-prone, at most partially successful, and ultimately infeasible.

References

[3] This approach is the standards-based version of migration, described below.

[4] The migration approach is often linked to the use of standards, but standards are not intrinsically a part of migration.

Next Previous

Return to CLIR Home Page >>

7. Criteria for an Ideal Solution

In contrast to the above strategies, an ideal approach should provide a single, extensible, long-term solution that can be designed once and for all and applied uniformly, automatically, and in synchrony (for example, at every future refresh cycle) to all types of documents and all media, with minimal human intervention. It should provide maximum leverage, in the sense that implementing it for any document type should make it usable for all document types. It should facilitate document management (cataloging, deaccessioning, and so forth) by associating human-readable labeling information and metadata with each document. It should retain as much as desired (and feasible) of the original functionality, look, and feel of each original document, while minimizing translation so as to minimize both labor and the potential for loss via corruption. If translation is unavoidable (as when translating labeling information), the approach should guarantee that this translation will be reversible, so that the original form can be recovered without loss.

The ideal approach should offer alternatives for levels of safety and quality, volume of storage, ease of access, and other attributes at varying costs, and it should allow these alternatives to be changed for a given document, type of document, or corpus at any time in the future. It should provide single-step access to all documents, without requiring multiple layers of encapsulation to be stripped away to access older documents, while allowing the contents of a digital document to be extracted for conversion into the current vernacular, without losing the original form of the document. It should offer up-front acceptance testing at accession time, to demonstrate that a given document will be accessible in the future. Finally, the only assumptions it should make about future computers are that they will be able to perform any computable function and (optionally) that they will be faster and/or cheaper to use than current computers.

8. The Emulation Solution

In light of the foregoing analysis, I propose that the best (if not the only) way to satisfy the above criteria is to somehow run a digital document's original software. This is the only reliable way to recreate a digital document's original functionality, look, and feel. The central idea of the approach I describe here is to enable the emulation of obsolete systems on future, unknown systems, so that a digital document's original software can be run in the future despite being obsolete. Though it may not be feasible to preserve every conceivable attribute of a digital document in this way, it should be possible to recreate the document's behavior as accurately as desired—and to test this accuracy in advance.

The implementation of this emulation approach would involve: (1) developing generalizable techniques for specifying emulators that will run on unknown future computers and that capture all of those attributes required to recreate the behavior of current and future digital documents; (2) developing techniques for saving—in human-readable form—the metadata needed to find, access, and recreate digital documents, so that emulation techniques can be used for preservation; and (3) developing techniques for encapsulating documents, their attendant metadata, software, and emulator specifications in ways that ensure their cohesion and prevent their corruption. Since this approach was first outlined (Michelson and Rothenberg 1992, Rothenberg 1995a), it has received considerable attention and has been cited as the only proposed approach that appears to offer a true solution to the problem of digital preservation (Erlandson 1996).

8.1 The right stuff

In principle, the proposed solution involves encapsulating three kinds of information with each digital document. In practice, there are a number of ways of doing this, some of which would be safer (but would use more storage), while others would involve somewhat more risk (but would use less storage). Figure 1 shows a logical view of this encapsulation. For clarity all items are shown explicitly, representing the logical model, although in practice, items that are required by many different documents might be stored in centralized repositories and pointed to by each document, rather than being replicated as part of each document.

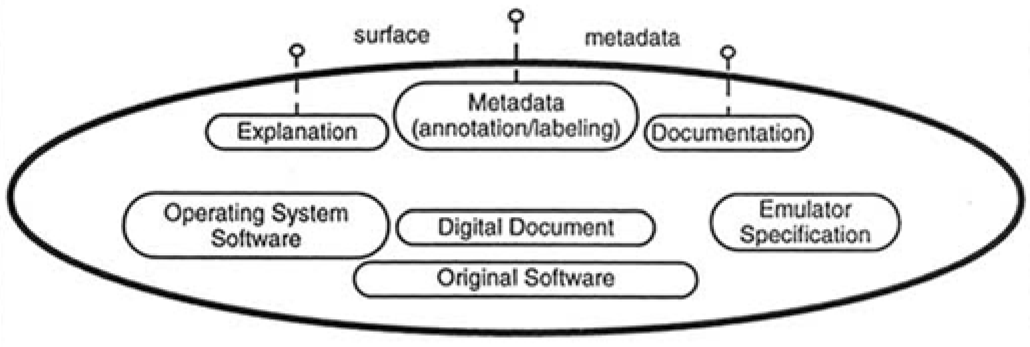

Figure 1: An encapsulated digital document

The first kind of information to be encapsulated comprises the document and its software environment. Central to the encapsulation is the digital document itself, consisting of one or more files representing the original bit stream of the document as it was stored and accessed by its original software. In addition, the encapsulation contains the original software for the document, itself stored as one or more files representing the original executable bit stream of the application program that created or displayed the document. A third set of files represents the bit streams of the operating system and any other software or data files comprising the software environment in which the document's original application software ran. It must be guaranteed that these bit streams will be copied verbatim when storage media are refreshed, to avoid corruption. This first group of encapsulated items represents the original document in its entire software context: given a computing platform capable of emulating the document's original hardware platform, this information should recreate the behavior of the original document.

The second type of information in the encapsulation of a document consists of a specification of an emulator for the document's original computing platform. The specification must provide sufficient information to allow an emulator to be created that will run on any conceivable computer (so long as the computer is capable of performing any computable function). This emulator specification cannot be an executable program, since it must be created without knowledge of the future computers on which it will run. Among other things, it must specify all attributes of the original hardware platform that are deemed relevant to recreating the behavior of the original document when its original software is run under emulation. Only one emulator specification need be developed for any given hardware platform: a copy of it (or pointer to it) can then be encapsulated with every document whose software uses that platform. This provides the key to running the software encapsulated with the document: assuming that the emulator specification is sufficient to produce a working emulator, the document can be read (accessed in its original form) by running its original software under this emulator.

The final type of information in the encapsulation of a document consists of explanatory material, labeling information, annotations, metadata about the document and its history, and documentation for the software and (emulated) hardware included in the encapsulation. This material must first explain to someone in the future how to use the items in the encapsulation to read the encapsulated digital document. In order to fulfill this function, at least the top level of this explanatory material must remain human-readable in the future, to serve as a "bootstrap" in the process of opening and using the encapsulation. This is one place where standards may find a niche in this approach: simple textual annotation standards (which might evolve over time) would provide one way of keeping explanatory material human-readable. If translation of this explanatory material is required to keep it human-readable (that is, if the annotation standards themselves evolve), the translation might be performed when the encapsulation is copied to new media: I refer to this limited form of translation as transliteration.[5] Any such translation must be reversible without loss, to ensure (and make it possible to verify) that the explanatory material is not corrupted. (These same techniques must be used to store emulator specifications, which must also remain human-readable in the future.) Additional metadata in the encapsulation describe the original document and provide labeling information that must accompany the document. Finally, additional metadata must provide historical context, provenance, life cycle history, and administrative information to help manage the document over time.

8.2 Annotate, Encapsulate, Transliterate and Emulate

Given a suitable emulator specification for a given obsolete hardware platform (which need only be created once for all documents whose software uses that platform), the process of preserving a digital document can be summarized as a sequence of four steps: annotate, encapsulate, transliterate and emulate. That is, (1) create any annotations needed to provide context for the document and to explain how to open and use the encapsulation; (2) encapsulate with the document all of the items described in the previous section; (3) when necessary (optionally, at each media refresh cycle), transliterate annotations to keep them human-readable; and (4) in the future, open the encapsulation, create the specified emulator, and run the emulator on a future computer. This allows the original software to be run under emulation, thereby recreating the saved document.

The sequence of events that must work in order for the emulation approach to allow an obsolete digital document to be read is illustrated in figure 2. The items in the top row of this figure represent elements that must be present for the scheme to work. Starting from the left, we must have a way of interpreting an emulator specification to produce a working hardware emulator (whether this interpretation is performed manually or automatically), and we must have a readable emulator specification for the required obsolete hardware (the original hardware and software are denoted HW and OS, respectively). This combination of a readable emulator specification and an interpreter for such specifications allows us to produce an emulator for the original hardware.

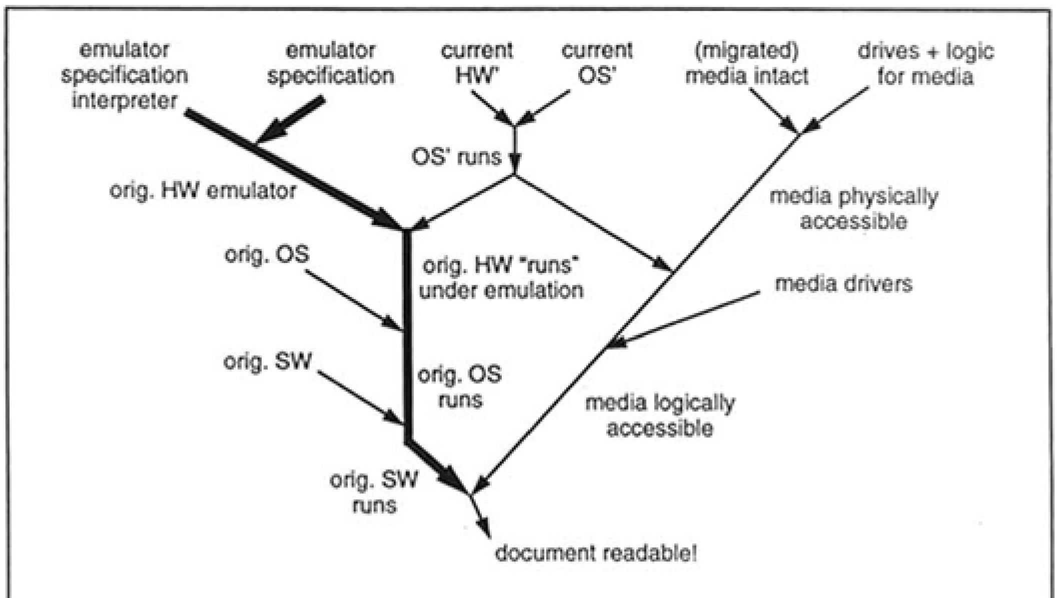

Figure 2: Using emulation to read an obsolete digital document

As shown in the middle of the top row, we then need a working, current computer and operating system (denoted HW' and OS') that can run the emulator: together, these produce a running OS' environment, which is required to support both the emulation branch (shown by heavy lines at the left of the figure) and the media-access branch (shown at right). Following the media-access branch down from the upper right, the obsolete digital document itself must also exist on some current storage medium (to which it will have presumably migrated from its original medium) for which physical drives and device software are available. Assuming we can run the necessary driver software for this medium under the current hardware/operating system environment (HW'/OS'), we can thereby access the bit stream of the original document. Finally, going back to the main, emulation branch, running the emulator of the original, obsolete hardware (HW) in the current HW'/OS' environment effectively "runs" the original hardware (under emulation); this allows us to run the original, saved (obsolete) operating system (OS), which in turn allows us to run the original, saved (obsolete) application software (SW) needed to read the saved (obsolete) digital document.

Though it may appear prohibitively inefficient to have to create and use an emulator to read each old document, three factors should be kept in mind. First, the inclusion of contextual annotation in the encapsulation makes it unnecessary to use emulation to perform routine management functions on the document, such as copying it, filing it, or distributing it. Emulation is needed only when the document is to be read or when its content is to be extracted for translation into some vernacular form.[6]

Second, an emulator specification for a given obsolete hardware platform need be created only once for all documents whose software uses that platform. This provides tremendous leverage: if an emulator specification is created for any document or document type, it will confer longevity on all other digital documents that use any of the software that runs on the given hardware platform.

Third, an emulator for a given obsolete platform need be created only once for each future platform on which emulation is required to run. Once created for each new generation of computer, the emulator for a given obsolete platform can be run whenever desired on any computer of that new generation. Generating new, running emulators for new computing platforms from saved emulator specifications will therefore be a rare process: once it has been done to access any document on a new platform, the resulting emulator for that platform can be used to access all other documents saved using the emulation scheme. The process of generating an emulator from its specifications can therefore be relatively inefficient (since it need be performed only infrequently), so long as the emulator that is generated is reasonably efficient when it runs.

8.3 Ancillary issues

Saving proprietary software, hardware specifications, and documentation, as required by this emulation strategy, raises potential intellectual property issues. Hardware specifications of the kind required for emulation are not necessarily proprietary, and since emulator specifications are not currently produced by hardware vendors (or anyone else), their intellectual ownership is as yet undefined. While application software and its documentation is often proprietary, the application programs required to access saved documents in general need be no more than readers for the desired document format, rather than editing programs. Such readers (along with their documentation) are often provided free by software vendors to encourage the use of their editing software. Operating system software and drivers, on the other hand, may very well be proprietary, and intellectual property restrictions or fees for these essential items must be respected if this approach is to work. Since the whole point of encapsulating this software is to make it available in the future, when it would otherwise be obsolete, one possible strategy would be to negotiate the free use of obsolete software, or to amend the copyright law to extend the principle of fair use to cover obsolete software.

For this strategy to work, responsibility for developing emulator specifications of the kind required would have to be accepted by one or more agencies, institutions, or market segments. Similarly, explanatory text standards would have to be developed and maintained, and responsibility would have to be accepted for refreshing media and performing transliteration (translating this explanatory material into new human-readable forms) when necessary.

8.4 Strengths and limitations of the emulation approach

It may appear that emulating a hardware platform simply to run application software is unnecessarily roundabout. If what is really desired is to emulate the behavior of the original digital document, why go to the trouble of running its original software at all? The answer to this is that we do not yet have any formal (or even informal) way of describing the full range of behaviors possible for even the simplest of digital documents, such as are produced by word processing programs. Describing the behavior of dynamic, interactive, hypermedia documents poses a far greater challenge. The only adequate specification of the behavior of a digital document is the one implicit in its interaction with its software. The only way to recreate the behavior of a digital document is to run its original software.

It may then be argued that instead of actually running original software, we might emulate the behavior of that software. This would provide considerable leverage over emulating the behavior of individual documents, since a given application program may be used for thousands (or millions) of different documents; emulating the behavior of that program would avoid having to understand and recreate the behavior of each individual document. However, we have no adequate way of specifying the behavior of most programs. The only meaningful specification of a program's behavior is implicit in its interaction with its underlying software/hardware environment; that is, programs are self-describing, but only when they run. The only way to tell what a program really does is to run it.

Alternatively, it might be argued that most application programs run under an operating system (though some may run on a "bare" machine), so we might emulate the behavior of the OS to provide a virtual platform for all applications that run under that OS. This would provide even greater leverage than would emulating applications, since many different applications run on a given OS. (Although some applications run on several different operating systems, there are many more application programs than there are operating systems.) Emulating an OS would avoid having to emulate all the applications that run on it. However, it is at least as difficult to emulate the behavior of an OS as it is to emulate the behavior of an application program; in fact, it is probably more difficult, since an OS interacts with every aspect of the computing environment, whereas most applications are far more constrained. So the argument against emulating an application applies a fortiori against emulating an OS. Nevertheless, I do not rule out this possibility: in some cases, it may be preferable to emulate a hardware platform along with an OS to produce a virtual hardware/software platform that can run application programs. The approach proposed here allows for this variation, though it assumes that emulating hardware platforms will usually make the most sense.

Emulating the underlying hardware platform appears to be the best approach, given the current state of the art. We do not have accurate, explicit specifications of software, but we do (and must) have such specifications for hardware: if we did not, we could not build hardware devices in the first place. Why is it that we can specify hardware but not software? Any specification is intended for some reader or interpreter. Application software is intended to be interpreted automatically by hardware to produce an ephemeral, virtual entity (the running application) whose behavior we do not require to be fully specified (except to the hardware that will run it), since it is intended to be used interactively by humans who can glean its behavior as they use it. On the other hand, a hardware specification is interpreted (whether by humans or software) to produce a physical entity (a computer) whose behavior must be well-specified, since we expect to use it as a building block in other hardware and software systems. Hardware specifications are by necessity far more rigorous and meaningful than those of software. Emulating hardware is therefore entirely feasible and is in fact done routinely.[7]

Hardware emulation is also relatively easy to validate: when programs intended for a given computer run successfully on an emulator of that computer, this provides reasonable assurance that the emulation is correct. Test suites of programs could be developed specifically for the purpose of validation, and an emulator specification could be tested by generating emulators for a range of different existing computers and by running the test suite on each emulator. A test suite of this kind could also be saved as part of the emulator specification and its documentation, allowing an emulator generated for a future computer to be validated (in the future, before being used) by running the saved test suite. In addition, the computer museum approach dismissed above might be used to verify future emulators by comparing their behavior with that of saved, obsolete machines.

Furthermore, of the potential emulation approaches discussed here, emulating hardware has the greatest leverage. Except for special-purpose embedded processors (such as those in toasters, automobiles, watches, and other products), computers are rarely built to run a single program: there are generally many more programs than hardware platforms, even though some programs may run on more than one platform. At any given moment, there are relatively few hardware platforms in existence, though new hardware platforms may appear with greater frequency than new operating systems or applications (unless we consider each version of an OS to be a different instance, which it really is). Emulating hardware obviates the need to emulate the behavior of operating systems, application programs, and individual digital documents, all of which are problematic; it therefore appears to be by far the most efficient as well as the most viable emulation approach.

Finally, hardware emulation is a well-understood, common technique. It has been used for decades, both to help design new hardware and to provide upward compatibility for users.

8.5 Natural experiments related to the emulation approach

Two classes of natural experiments suggest that the emulation approach described here should work. Though none of these experiments addresses all of the questions that must be answered in order to use emulation as a basis for digital preservation, they show that key pieces of the strategy have worked in the past.

The first class consists of examples of bundling digital documents with their original software to ensure that they are accessible and readable. For example, Apple Macintosh software is often distributed with a README file that describes the software, explains how to install it, and gives other information such as restrictions on use and information about bugs. (This is also an example of encapsulating explanatory annotation with software.) In order to ensure that the README file will be readable by any user, distribution disks typically include a copy of a simple text-editing program (SimpleText) that can display the README file. Though most users already have at least one copy of SimpleText on their systems, as well as other, more powerful editors, most software vendors prefer not to assume that this will be the case. In the emulation approach, digital documents would be bundled with their original software, just as the README file is bundled with software capable of reading it in this example.

A second example of bundling occurs in the PC world, involving the compression scheme called PKZIP. When a file is compressed using this software, a decompression program, such as PKUNZIP, is required to expand the file. However, an option in PKZIP allows a simple version of an unzip program to be bundled with each compressed file. Choosing this option creates an executable file which, when run, expands automatically to the original file, avoiding the issue of whether the recipient of a compressed file will have the appropriate decompression software on hand.

A final example is a collection of images distributed on the Planetary Data Systems CD-ROM from NASA's Jet Propulsion Laboratory (JPL). The imagery on this CD is designed to be read with Adobe Reader 2, a free program that can display files encoded in Adobe's popular portable document format (PDF). If the user tries to display these images with a later version of Adobe Reader, the images refuse to display themselves: not only are the images bundled with appropriate software, but they are protected from being accessed by any other software, since there is no guarantee that such software will treat the images appropriately. In this case, not even later versions of the same program (such as Adobe Reader 3) are allowed to read the images; though this may seem restrictive, it is in fact a good approach, since later versions of software do not always treat files created by older versions appropriately.

All of these examples bundle software to be run on a known platform, so none of them provides much longevity for their documents. Nevertheless they do prove that bundling original software with a document is an effective way of making sure that the document can be read.

The second class of natural experiments involves the use of emulation to add longevity to programs and their documents. The first example is a decades-old practice that hardware vendors have used to provide upward compatibility for their customers. Forcing users to rewrite all of their application software (and its attendant databases, documents, and other files) when switching to a new computer would make it hard for vendors to sell new machines. Many vendors (in particular, IBM) have therefore often supplied emulation modes for older machines in their new machines. The IBM 360, for example, included an emulation mode for the older 7090/94 so that old programs could still be run. Apple did something similar when switching from the Motorola 68000 processor series to the PowerPC by including an emulator for 68000 code; not only did this allow users to run all of their old programs on the new machine, but significant pieces of the Macintosh operating system itself were also run under emulation after the switch, to avoid having to rewrite them. Whether emulation is provided by a special mode using microcode or by a separate application program, such examples prove that emulation can be used to keep programs (and their documents) usable long after they would otherwise have become obsolete.

A second example of the use of emulation is in designing new computing platforms. Emulation has long been used as a way of refining new hardware designs, testing and evaluating them, and even beginning to develop software for them before they have been built. Emulators of this kind might be a first step toward producing the emulator specifications needed for the approach proposed here: hardware vendors might be induced to turn their hardware-design emulators into products that could satisfy the emulator scheme's need for emulator specifications.

A final example of the use of emulation is in the highly active "retro-computing" community, whose members delight in creating emulators for obsolete video game platforms and other old computers. There are numerous World Wide Web sites listing hundreds of free emulators of this kind that have been written to allow old programs to be run on modern computers. A particularly interesting example of this phenomenon is the MAME (Multiple Arcade Machine Emulator) system, which supports emulation of a large number of different platforms, suggesting that emulation can be cost-effective for a wide range of uses.

These three examples consist of emulators that run on existing hardware platforms, so they do not address the problem of specifying an emulator for a future, unknown computer; but they prove that emulation is an effective way of running otherwise obsolete software.

References

[5] While transliteration need not be tied to refresh cycles, doing so minimizes the number of passes that must be made through a collection of digital material. If a single annotation standard is selected for all documents in a given corpus or repository during a given epoch to simplify document management, transliteration could be performed for all documents in a collection in lock-step, just as media refreshing is done in lock-step. Though transliteration does not necessarily have to be done at the same time as refreshing, doing so would be more efficient (though potentially riskier) than performing transliteration and refreshing at different times.

[6] The emulation environment must be designed to allow such extraction in order to facilitate the generation of vernacular versions of obsolete documents.

[7] It is not clear whether digital preservation needs to include the retention of attributes of the original medium on which a digital document was stored. It can be argued that digital documents are (or should be) logically independent of their storage media and therefore need not preserve the behavior of these media; however, a counterarugment to this might be that since some digital documents are tailored to specific media (such as CD-ROM), they should retain at least some attributes of those media (such as speed). The approach described here is neutral with respect to this issue: it allows attributes of storage media to be emulated when desired but does not require them to be.

Next Previous

Return to CLIR Home Page >>

9. Research Required for the Emulation Approach

In order to prove the feasibility of the emulation approach, research is required in three areas: (1) techniques must be developed for specifying emulators that will run on unknown, future computers; (2) techniques must be developed for keeping annotations and explanations human-readable in the future; and (3) techniques must be developed for encapsulating documents, software, emulator specifications, and associated annotations and metadata to ensure their mutual cohesion and prevent their corruption.

9.1 Emulator specification formalism

An emulator specification formalism must be developed that captures all relevant attributes of a hardware platform, including interaction modes, speed (of execution, display, access, and so forth), display attributes (pixel size and shape, color, dimensionality, and so forth), time and calendar representations, device and peripheral characteristics, distribution and networking features, multiuser aspects, version and configuration information, and other attributes. The formalism must be extensible so that future attributes can be added when needed (for example, for compliance with future Y10K standards). The set of attributes needed to ensure that a future emulation precisely reproduces an obsolete platform in all possible aspects is unbounded, but the scheme assumes that a certain degree of variance in the behavior of emulators will be acceptable. (This variance corresponds to that in the original program's behavior when executed on different contemporary systems and configurations, using different monitors, keyboards, disk drives, and other peripheral devices.)

Emulator specifications must be saved so that they can be used effectively to produce emulators in the future. There are several possible ways of doing this. First, an abstract, formal description could be saved, which could be interpreted by a human or program in the future to enable construction of the desired emulator. Second, an executable description (that is, an emulator program written in a high-level language) could be saved, which would be designed to run on some simple abstract machine that could easily be implemented in the future; instructions for implementing that abstract machine and/or an abstract formal description of it would be saved along with the emulator to allow it to run on future computers. Alternatively, the second approach could be transformed into an instance of the first by making the executable description abstract and formal, thereby allowing it either to be interpreted by a human or program in the future or to be run on a future implementation of an abstract machine. All of these alternatives require storing formal and/or explanatory descriptions that can be used to bootstrap the process of creating a future emulator. To ensure that these descriptions remain human-readable in the future, they must be stored in the annotation form discussed next.

9.2 Human-readable annotations and explanations

Ideally, the emulation scheme would be self-describing: that is, a suitable program running on a future computer, when asked to access an obsolete document saved using this scheme, would automatically interpret the saved explanations to find out how to open the encapsulation, generate the required emulator (or find one that has already been generated for this type of computer), and run the document's saved software under this emulator to access the document itself. Alternatively, a user could interpret the saved explanations to perform the same steps. In either case, the key to successfully accessing a document saved using the emulation approach lies in the saved explanations that accompany the document, including explanations of how to use the encapsulation itself, user documentation, version and configuration information for all the software that is to be run under emulation (and for the emulated hardware), and the emulator specification itself. Whether or not these saved explanations can be automatically interpreted by future computer programs, they *must* remain readable by future humans, to ensure that saved documents are not lost.

The emulation approach requires the development of an annotation scheme that can save these explanations in a form that will remain human-readable, along with metadata which provide the historical, evidential, and administrative context for preserving digital documents. There has been considerable work in the library, archives, scientific data, and records communities on identifying such metadata (Cox 1994 and 1996, IEEE 1997, NRC 1995, Rothenberg 1996).

Future users of digital documents preserved using the emulation approach will be faced with an encapsulated collection of components that need to be used in a particular way in order to read the desired document. First and foremost, users must be able to read some intelligible explanation that tells them how to proceed. This explanation must itself be a digital document (if only to guarantee that it accompanies the other components of the encapsulation), but it must be human-readable if it is to serve its purpose. It will generally be of the same vintage as the encapsulated digital document whose exhumation it explains, but it cannot be stored in the same way as that document, or it will be equally unreadable. The solution to this conundrum lies in restricting the form of this explanatory documentation, for example, to simple text (or possibly text plus simple line drawings).

Even if the encoding of this explanatory material is standardized, however, whatever standard is chosen will eventually become obsolete, which is why the emulation strategy allows annotations and explanations to be translated (transliterated) whenever necessary. In order to guarantee that this translation is performed without loss, we must develop subset-translatable encodings, which I define as having the property that if some encoding Y is subset-translatable into another encoding Z, then anything expressed in Y can be translated into a subset Zy of Z, and anything in the resulting subset Zy can be translated back into Y without loss. This allows Z to be a proper superset of Y (not limited to Y's expressivity) while ensuring that anything that is expressed in Y can be translated into Z and back into Y again without loss. A sequence of such encodings, evolving as necessary over time, will solve the readability problem for annotations: each encoding in this sequence serves as an annotation standard during a given epoch.

Although it is logically sufficient—having asserted that an encoding Y is subset-translatable into encoding Z—to translate a document from Y to Z and discard the original Y-form of the document, this is unlikely to convince skeptical future users. It is therefore also important to develop the concept of a subset-translator (consisting in each case of a table or a process, depending on the complexity of the translation) that shows how to translate Y into Z and back again. If this translator is saved, along with definitions of encodings Y and Z, and the Y and Z forms of all translated information, then any future user can verify that Y is indeed subset-translatable into Z, that the information was correctly translated from Y to Z, and that nothing was lost in this translation (by verifying that the reverse translation reproduces the original, saved Y-form of the information).[8] In order for all of this saved information (encodings, translators, history of translations that have been performed, and so forth) to remain readable in the future, it must be stored using this same transliteration scheme, that is, it must be encoded in a current annotation standard, to be subset-translated as needed in the future.

9.3 Encapsulation techniques

One final piece of the puzzle is required to make the emulation approach work: how do we encapsulate all of the required items so that they do not become separated or corrupted and so that they can be handled as a single unit for purposes of data management, copying to new media, and the like? While encapsulation is one of the core concepts of computer science, the term carries a misleading connotation of safety and permanence in the current context. An encapsulation is, after all, nothing more than a logical grouping of items. For example, whether these are stored contiguously depends on the details of the storage medium in use at any given time. The logical shell implied by the term *encapsulation* has no physical reality (unless it is implemented as a hardened physical storage device). And while it is easy to mark certain bit streams as inviolate, it may be impossible to prevent them from being corrupted in the face of arbitrary digital manipulation, copying, and transformation.

Techniques must therefore be developed for protecting encapsulated documents and detecting and reporting (or correcting) any violations of their encapsulation. In addition, criteria must be defined for the explanatory information that must be visible *outside* an encapsulation to allow the encapsulation to be interpreted properly.

Many encapsulated digital documents from a given epoch will logically contain common items, including emulator specifications for common hardware platforms, common operating system and application code files, software and hardware documentation, and specifications of common annotation standards and their translators. Physically copying all of these common elements into each encapsulation would be highly redundant and wasteful of storage. If trustworthy repositories for such items can be established (by libraries, archives, government agencies, commercial consortia, or other organizations), then each encapsulation could simply contain a pointer to the required item (or its name and identifying information, along with a list of alternative places where it might be found). Different alternatives for storing common items may appeal to different institutions in different situations, so a range of such alternatives should be identified and analyzed.

There is also the question of what should go inside an encapsulation versus what should be presented at its surface to allow it to be manipulated effectively and efficiently. In principle, the surface of an encapsulation should present indexing and cataloging information to aid in storing and finding the encapsulated document, a description of the form and content of the encapsulated document and its associated items to allow the encapsulation to be opened, contextual and historical information to help a potential user (or document manager) evaluate the relevance and validity of the document, and management information to help track usage and facilitate retention and other management decisions. All of this information should be readable without opening the encapsulation, since none of it actually requires reading the encapsulated document itself.

It is logically necessary only that the tip of this information protrude through the encapsulation: there must be some explanatory annotation on the surface that tells a reader how to open at least enough of the encapsulation to access further explanatory information inside the encapsulation. Even this surface annotation will generally not be immediately human-readable, if the encapsulation is stored digitally. If it happens to be stored on a physical medium that is easily accessible by humans (such as a disk), then this surface annotation might be rendered as a human-readable label on the physical exterior of the storage unit, but this may not be feasible. For example, if a large number of encapsulations are stored on a single unit, it may be impossible to squeeze all of their surface annotations onto the label of the unit. So in general, even this surface annotation will be on a purely logical surface that has no physical correlate. The reader of this surface annotation will therefore be a program rather than a human, though it may quickly deliver what it reads to a human. It must therefore be decided how such surface annotations should be encoded, for example, whether the annotation standards described above are sufficient for this purpose or whether a hierarchy of such standards—corresponding to different levels of immediate human-readability—should be developed.

10. Summary

The long-term digital preservation problem calls for a long-lived solution that does not require continual heroic effort or repeated invention of new approaches every time formats, software or hardware paradigms, document types, or recordkeeping practices change. This approach must be extensible, since we cannot predict future changes, and it must not require labor-intensive translation or examination of individual documents. It must handle current and future documents of unknown type in a uniform way, while being capable of evolving as necessary. Furthermore, it should allow flexible choices and tradeoffs among priorities such as access, fidelity, and ease of document management.

Most approaches that have been suggested as solutions to this problem —including reliance on standards and the migration of digital material into new forms as required—suffer from serious inadequacies. In contrast, the emulation strategy as elaborated above, though it requires further research and proof of feasibility, appears to have many conceptual advantages over the other approaches suggested and is offered as a promising candidate for a solution to the problem of preserving digital material far into the future.

References

[8] Although a sequence of such translations may be needed over time, all that is really required is to save the sequence of encodings and translators: future custodians of this explanatory information could then safely defer translating a particular annotation until it is needed, so long as its encoding is not lost.

Next Previous

Return to CLIR Home Page >>

REFERENCES

Bearman, D. 1993. "The Implications of *Armstrong v. Executive Office of the President* for the Archival Management of Electronic Records." *American Archivist* 56: 150–60.

Bearman, D. (ed.). 1991. *Hypermedia and Interactivity in Museums: Proceedings of an International Conference*. Pittsburgh.

Bearman, D. 1992. "Documenting Documentation." *Archivaria* 34 (Summer).

Bikson, T. K. Forthcoming. "Managing Digital Documents: Technology Challenges and Institutional Responses." *In Proceedings of the Internal Conference of the Round Table on Archives*. Stockholm, Sweden: International Council on Archives (September 1998).

———. 1997. *Strategies for Implementing Document Management Technology*. ACC/1997/ISCC/4. Geneva: United Nations Information Systems Coordination Committee (ISCC), Annex II.

———. 1994. "Organizational Trends and Electronic Media." *American Archivist* 57(1): 48–68. (Also available from RAND as Reprint RP-307.)

Bikson, T. K., and E. J. Frinking. 1993. "Preserving the Present: Toward Viable Electronic Records." Den Haag: Sdu Publishers. (Parts of this book are available as RAND Reprint RP-257.)

Bikson, T. K., and S. A. Law. 1993. "Electronic Information Media and Records Management Methods: A Survey of Practices in United Nations Organizations." *Information Society* 9(2): 125–44. (Also available from RAND as Reprint RP-508.)

Codd, E. F. 1982. "Relational Database: A Practical Foundation for Productivity." *CACM* 25(2): 109–17.

Coleman, J., and Don Willis. 1997. *SGML as a Framework for Digital Preservation and Access*. Washington, D.C.: Commission on Preservation and Access.

Cox, Richard J. 1994. "Re-Discovering the Archival Mission: The Recordkeeping Functional Requirements Project at the University of Pittsburgh, A Progress Report." *Archives and Museum Informatics* 8(4): 279–300.

———. 1996. "The Record in the Information Age: A Progress Report on Reflection and Research." *Records & Retrieval Report* 12(1): 1–16.

Day, Michael. 1997. "Extending Metadata for Digital Preservation." *Ariadne*. Available from http://www.ariadne.ac.uk/issue9/metadata/ (May).

Dollar, C. M. 1992. *Archival Theory and Information Technologies: The Impact of Information Technologies on Archival Principles and Practices, Information and Documentation*. Series 1, edited by Oddo Bucci. Macerata, Italy: University of Macerata.

Dublin Core Metadata Initiative. 1998. "The Dublin Core: A Simple Content Description Model for Electronic Resources." Sponsors: Online Computer Library Center, The Library of Congress, Coalition for Networked Information, National Science Foundation, National Libraries of Australia and Finland, National Computational Science Alliance, Distributed Systems Technology Centre (Australia), and UK Office for Library and Information Networking. Available from http://purl.org/dc.

Erlandsson, Alf. 1994. "The Principle of Provenance and the Concept of Records Creator and Record: Legal Development." In *The Principle of Provenance,* pp. 33–49. Stockholm: Swedish National Archives.

———. 1996. *Electronic Records Management: A Literature Review.* International Council on Archives' (ICA) Study. Available from http://www.archives.ca/ica. ISBN 0-9682361-2-X.

Fonseca, F., P. Polles, and M. Almeida. 1996. *Analysis of Electronic Files of Bank Reports*. Washington. D.C.: World Bank, Business Unit Document Services.

Fox, E. A., and G. Marchionini (guest eds.). 1998. "Toward a World Wide Digital Library." *CACM* 41(4): 28–98.

Getty Conservation Institute, Getty Information Institute, Long Now Foundation. 1998. "Time & Bits: Managing Digital Continuity." Available from http://www.ahip.getty.edu/timeandbits/intro.html.

Giguere, M. 1996. "Metadata Attribute Recommendations." In *Human Resources Information System—RFP*. Philadelphia, Pa.: City of Philadelphia. Available from http://www.lis.pitt.edu/~nhprc/xhibit4.html.

Hedstrom, Margaret. 1991. "Understanding Electronic Incunabula: A Framework for Research on Electronic Records." *American Archivist* 54(3): 334–54.

———. 1993. "Electronic Records Program Strategies: An Assessment." In *Electronic Records Management Program Strategies,* edited by Margaret Hedstrom. Archives and Museum Informatics. Pittsburgh, Pa.

Horsman, Peter. 1994. "Taming the Elephant: An Orthodox Approach to the Principle of Provenance." In *The Principle of Provenance*, pp. 51–63. Stockholm: Swedish National Archives.

Institute of Electrical and Electronics Engineers. 1997. *Proceedings of the Second IEEE Metadata Conference*. Silver Spring, Md., September 16–17.

Indiana University Electronic Records Project. 1997. "A Methodology for Evaluating Recordkeeping Systems." Draft (March).

Kenney, Anne R. 1997. *Digital to Microfilm Conversion: A Demonstration Project 1994–96*. Final report to the National Endowment for the Humanities. Ithaca, N. Y.: Cornell University Library. Available from http://www.library.cornell.edu/preservation/com/comfin.html.

Lancaster, F. W. 1986. *Vocabulary Control for Information Retrieval*.

Arlington, Va.: Information Resources Press.

Lesk, Michael. 1995. "Preserving Digital Objects: Recurrent Needs and Challenges." Available from http://www.lesk.com/mlesk/auspres/aus.html.

———. 1992. *Preservation of New Technology: A Report of the Technology Assessment Advisory Committee to the Commission on Preservation and Access*. Washington, D.C.: Commission on Preservation and Access.

Manes, S. 1998. "Time and Technology Threaten Digital Archives." *New York Times,* April 7, *Science Times* section, p. F-4.

Michelson, A., and Jeff Rothenberg. 1992. "Scholarly Communication and Information Technology: Exploring the Impact of Changes in the Research Process on Archives." *American Archivist* 55(2).

Morelli, Jeffrey D. 1998. "Defining Electronic Records: Problems of Terminology." In *History and Electronic Artefacts,* edited by Edward Higgs, pp. 169–83. Oxford: Clarendon Press.

Morris, R. J. 1998. "Electronic Documents and the History of the Late Twentieth Century: Black Holes or Warehouses." In *History and Electronic Artefacts,* edited by Edward Higgs, pp. 31–38. Oxford: Clarendon Press.

National Archives and Records Administration (NARA). 1991. *Management, Preservation and Access for Electronic Records with Enduring Value*. Washington, D. C.: NARA. July 1.

National Research Council. 1995. *Preserving Scientific Data on our Physical Universe: A New Strategy for Archiving the Nation's Scientific Information Resources*. Washington, D.C.: National Academy Press.

Popkin, J., and A. Cushman. 1993. *Integrated Document Management —Controlling a Rising Tide*. Stamford, Conn.: Gartner Group.

Roberts, D. 1994. "Defining Electronic Records. Documents and Data." *Archives and Manuscripts* 22(2): 14–26.

Rothenberg, Jeff. 1997. "Digital Information Lasts Forever—Or Five Years, Whichever Comes First." RAND Video V-079.

———. 1996. "Metadata to Support Data Quality and Longevity." First IEEE Metadata Conference, April 16–18. Silver Spring, Md. Available from http://computer.org/conferen/meta96/rothenberg_paper/ieee.data-quality.html.

———. 1995a. "Ensuring the Longevity of Digital Documents." *Scientific American* 272(1): 24–29.

———. 1995b. "Survival of the Digits." *Communicating Business*, Issue 4 (Spring). London: Forward Publishing.

Sanders, Terry. 1997. *Into the Future*. Video/film (60-minute version shown on PBS; 30-minute version also available). American Film Foundation, 1333 Ocean Ave., Santa Monica, Calif. 90406.

Schurer, Kevin. 1998. "The Implications of Information Technology for the Future Study of History." In *History and Electronic Artefacts*, edited by Edward Higgs, pp. 155–58. Oxford: Clarendon Press.

Smith, Abby. 1998. "Preservation in the Future Tense." *CLIR Issues 3* (May/June). Washington, D.C.: Council on Library & Information Resources.

Swade, Doron. 1998. "Preserving Software in an Object-Centred Culture." In *History and Electronic Artefacts*, edited by Edward Higgs, pp. 195–206. Oxford: Clarendon Press.

Thibodeau, Kenneth. 1991. "To Be Or Not to Be: Archives for Electronic Records." In *Archival Management of Electronic Records*, edited by David Bearman. Pittsburgh, Pa.: Archives and Museum Informatics.

United Nations Advisory Committee for the Coordination of Information Systems (ACCIS). 1990. *Management of Electronic Records: Issues and Guidelines*. New York: United Nations.

———. 1992. *Strategic Issues for Electronic Records Management: Toward Open System Interconnection*. New York: United Nations.

United States Congress. 1990. House Committee on Government Operations. *Taking a Byte out of History: The Archival Presentation of Federal Computer Records*. House Report 101–987. Washington, D.C.

United States District Court for the District of Columbia. 1993. Opinion of Charles R. Richey, U.S. District Judge, January 6.

Van Bogart, John W. C. 1996. "Long-Term Preservation of Digital Materials." Paper presented at the National Preservation Office Conference on Preservation and Digitisation: Principles, Practice and Policies. University of York, England, September 3–5.

Waters, Donald, and John Garret. 1996. *Preserving Digital Information*. Report of the Task Force on Archiving of Digital Information. Commissioned by the Commission on Preservation and Access and the Research Libraries Group, Inc. (May 1).

Previous

Return to CLIR Home Page >>