Preserving Digital Documents for the Long-Term

Raymond A. Lorie
IBM Almaden Research Center
San Jose, California, USA

Abstract

As more and more documents are created and archived digitally, their preservation is becoming critical. A digital document is a sequence of bits that needs to be interpreted and the interpretation process must be archived with the document. The process may be simple or very complex, and various methods have been proposed, depending upon the required functionality.

The paper reviews the most popular methods, with particular emphasis on the use of a "Universal Virtual Machine" (or UVC).[1,2] The UVC method consists of archiving with the data a program P that dynamically converts the internal format into an easily readable structure which identifies the various elements (much like XML does) down to a point where it becomes much easier to define the rest of the interpretation process. In the future, an interpreter of the UVC program will enable the execution of P on any computer. The focus of the paper is primarily on archiving printable documents which require both content and presentation to be preserved; a short section will briefly cover more general documents.

The paper makes reference to an existing prototype and how it was used in some proof of concept projects.

Introduction

The problem of archiving digital information for the long-term is receiving increasing attention. The technical challenge is twofold. First, the bits must be preserved. Second, the future client must be able to interpret the bits and build a modern viewer or any application that needs to access the data.

The sub-problem that preserves the bits for the long-term presents interesting challenges but they are not insurmountable. The business-as-usual solution consisting of copying the information periodically from one medium/device combination to another one, works. It faithfully preserves the bits and always takes advantage of the latest storage technology. This is important since it reduces the requirement for shelf space and real estate.

On the other hand, preserving the interpretation of the bits is a different story. Let us consider three cases:

Case 1: **On the Side of Simplicity**

Consider a black and white image stored as an uncompressed bit map. In addition to archiving the bits, a certain amount of metadata is needed to describe how to interpret them. It contains the identification of the type ("bitmap") and two values, height and width. The metadata associate with "bitmap" would explain in English that the pixels are stored line by line, starting at the top/left, one bit per pixel (1 for black), etc. There is no process to be saved; the future client will be able to write a modern viewer.

Case 2: **On the Side of Complexity**

Consider an interactive educational CD, and assume that an archivist wants to be sure that historians will be able to play the module in fifty years from now. The CD file(s) can be stored as a binary file, and, in order to preserve the program functionality, the metadata must include enough information to re-enact the behavior of the 2003 computer in the future. This can hardly be described in English; so the best bet is to include in the metadata a platform independent version of an emulator of the 2003 computer (plus a user's guide for the program). Building an emulator is hard, even more so if timings must be handled accurately.

Case 3: **Preserving a Printable Document**

On the complexity scale, the case falls between the two extreme cases 1 and 2. It is more complex than case 1 because the data structure is much more complex, and because a certain process, tightly bound to the data must be applied to the data to make it meaningful. That process is much too difficult to explain in English and suggests the use of a program. For preservation, such a program needs to be specified in a way that will ensure its longevity across multiple changes in machine architecture and software.

Preservation Methods

This paper focuses on archiving printable documents, preserving their content and presentation. The most important approaches are now summarized.

Conversion

It is the most obvious, business-as-usual, method. When a new system is installed, it coexists with the old one for some time, and all files are copied from one system to the other. If some file formats are not supported by the new system, the files are converted to new formats and applications are changed accordingly. However, for long-term preservation of rarely accessed documents, conversion becomes an unnecessary burden; it is also prone to errors. Program conversion may be reasonable in some cases but data conversion is generally not.

Emulation

In Ref. [3], Jeff Rothenberg proposes to save, together with the data, the program that was used to create/manipulate the information in the first place. Then, the only way to decode the bit stream is to run that old program; this always requires an interpreter of the real machine. References [3] and [4] suggest different methods for defining how an interpreter can be specified; but the feasibility of these methods has not been proven. The emulation approach suffers from two other drawbacks.

1. Saving the original program is justifiable for re-enacting the behavior of a program, but it is overkill for data archiving. In order to archive data created online, it is hardly necessary to be able to re-enact the behavior of a full interactive system.
2. The original program generally shows the data (and more often, results derived from the data) in one particular output form. The program does not make the data accessible; it is therefore impossible to export the original data from the old system to the new one.

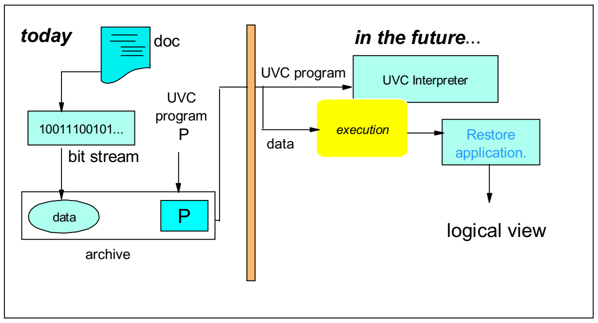

Figure 1. The UVC preservation mechanism

Reliance on a Virtual Machine

A few definitions are in order:

- internal format: it is the way the information is stored in the original bit stream representing the document.
- logical view: it is the set of logical data elements (and the connections between them) that are contained in the internal format (or obtained by computation), individually isolated and tagged with a semantic tag (very much like XML).
- derived internal format: it is another internal format which preserves all the *essential information* of the original document. The *essential information* means anything that has a value for the archivist. In some cases it may be the ASCII text of a document; in other cases, it may be the full behavior of a program.

Today, an application program generates a data file, which is archived for the future (see Fig. 1). In order for the file to be understood in the future, a program P is also archived, that can decode the internal format and present the data to the client in a logical view, much easier to understand. To avoid the obsolescence problem, that program P is written for a UVC machine.

In the future, a restore application program reads the bit stream and passes P to a UVC interpreter, which executes it. During the execution, the data is decoded and returned, element by element, according to the logical view. Note that the logical view is chosen at the time the UVC program is written. The program extracts the various data elements and labels them with the appropriate tags. Nothing impedes the UVC program to perform other operations, in addition to the extraction of data elements, and even inserting the results in the logical view. The logical view is XML-like because it returns tag-element pairs organized hierarchically like XML, but results of simple API invocations and not as a character string that must be parsed. It is worth reiterating that the logical view is generated at access time, by the UVC program, from the internal format stored in the archive.

In order to understand the data, the future client must know the schema. The definition of the schema is just another set of data. The schema is also archived with the file, together with a UVC program that decodes it and returns its data elements according to a "schema to read schemas". That schema is fixed; its predefined structure and tags are part of the UVC Convention,[2] which must be universally known, now and in the future. The Convention does not need to be stored with each object but needs to be replicated in many places so that it remains accessible by anybody, anywhere and at any time.

A short parenthesis on the UVC: the specification of the UVC is summarized in Ref. [2]. Its architecture has been influenced by the fact that the machine does not need to be implemented physically. Therefore, there is no actual physical cost. For example, the UVC can have a large number of registers; each register has a variable number of bits plus a sign bit; the memory is unlimited in size, segmented, bit-oriented. The UVC has a small set of instructions, thus reducing the amount of work involved in developing an interpreter of the UVC instructions on a future real machine. An initial design of the UVC exists, with a corresponding interpreter. As mentioned later in this paper, two proofs of concept have been conducted.

Preserving Content and Presentation

This section focuses on printable documents (case 3), and considers various strategies.

Using Metadata

If the format is simple enough, all pertinent information can be described as an unstructured English text or as some easy to understand XML structure. A black and white image stored as a simple bitmap could be archived this way. The metadata would specify the width and height of the image in pixels, and explain that a bitmap is stored 8 pixels to a byte, by row, top to bottom, left to right, that bytes are padded at the end of a line, etc. The information must be complete so that a viewer can be written at any time.

Converting Once to XML

XML is often seen as a solution to the long-term preservation problem. This comes from the fact that an XML structure is much more understandable than any other representation. Actually, many document formats can already be converted to an XML representation (for example, the MS Office formats). However, an XML document must be accompanied by a fair amount of metadata describing what the tags mean and what kind of process is needed to convert the stored data into a meaningful visual output. If the process is complex, it cannot be described as an English text; it will instead require a program specification. Consider an Excel spreadsheet. The derived XML file contains the coordinates of each cell and the text that it contains. The text in a cell may be divided in various sections depending on typesetting attributes such as font, size, thickness, etc. This does not say how the text is split in multiple lines or where the individual characters are positioned on the line, and certainly not the shape of each character. It is therefore possible to generate an output that is fairly close to the original, but not to provide an exact replicate; the method provides a reduced level of functionality. The level of functionality can be raised by going one step down in the XML description. Instead of providing coordinates at the cell level, the description may include coordinates for every character, and the shape of every character may be represented as a small bit map. What is archived is then a "frozen" but faithful representation.

Archiving the Original Format of the Document

Although XML may play an important role in preservation, it is not a complete solution. In many cases, it may yield an easily understandable logical view but may be inappropriate as a storage format. And, it cannot be used to specify a process. This is where the UVC comes in., as illustrated in the following two example.

JPEG

Suppose you need to archive a JPEG image. We could argue that the format is so widespread that programs to decode it will always be available. But, again, this may not be the case; and anyway, there are hundreds of image formats that are in need of preservation support; we just use JPEG as an example.[5]

Here, it makes no sense to convert the image to an XML that is simple to decode since the main advantage of JPEG is precisely its complex encoding used for compression. The ideal is to store the JPEG representation but "see" an XML representation when retrieved from the archive. This is exactly what the UVC technique accomplishes. It consists of archiving the JPEG bit stream, accompanied by a UVC program to decode the JPEG internal structure and produce dynamically, on demand, a logical view. This view may be a simple hierarchy of tagged data elements: the number of lines N, the width in pixels W, a sequence of N lines where each line is a sequence of 24 bits words.

In the future, a Restore program will read the different data elements with their tags and do with them whatever is necessary; in particular, it can recreate the picture in an obvious way.

The method offers complete flexibility on the choice of the logical schema that would be the most appropriate. Actually, there is in general no reason why several different schemas would be needed for the same logical document type (here, "picture"). For presentation, a TIFF[5] image could be logically described with the same logical view. The big advantage is that a single program can be used later to view the image, independently of its original format

PDF

The method shown for an image can be generalized for documents with content and preservation. Consider the case of a PDF document.[6] PDF is increasingly seen as a candidate for a standard to support preservation of printable material. PDF captures well all aspects of the presentation, but documents are generally created and updated outside of the PDF world and then converted into PDF. Although it is hard to prove that the result of such a conversion is always faithful, the disparities are few. Thus, the PDF version, if verified by the producer, could be declared to be the original. (PDF as a commercial product poses some problems; they are addressed in the current work on a standard version called PDF/A.) The problem of preserving PDF (or PDF/A) is therefore of great interest.

One possible strategy is to follow a method similar to the one proposed for JPEG: keeping the original PDF file as the archive, and rely on a UVC program P, associated with the PDF type, to decode the format and produce a logical XML-like view form which can be easily understood (assuming of course that the needed metadata exists). Two options are possible. The first one is to freeze the presentation as done for the Excel example above. The second option is to capitalize on the fact that the UVC can describe any process. So, even the process that generates the shape of the characters dynamically, based on some input parameters, can be used. Of course, the writing of the UVC program becomes more difficult, but may be developed by simply compiling some of the original code of any system that renders PDF in all its details.

Using a Derived Internal Format

Archiving the data in its original format is certainly a good strategy. There are cases, however, where this strategy may be impracticable. An alternative is to convert a document from its original format F into a derived format F' at archival time. Then, the new format may be less optimized for a particular capability (such a generating the bitmap for printing) but more extensible to support other functions.

In order to build the derived format, the information from the original format must first be extracted; this may still require full knowledge of the original format. But it is often possible to simplify the task by using accessibility tools that exist today and may not be available in the future. In a joint project between IBM Research and the Koninklijke Bibliotheek in The Hague, Netherlands,[7] such a method was used to implement a preservation method for PDF documents. It starts by generating a file containing the full image of the page (using Ghostscript[8]), extracting the descriptive fields from the PDF file itself, and getting the text, its attributes and the bookmarks, element by element, from an HTML equivalent (produced by a tool from BCL[9]). The HTML file provides a tagged element for every piece of homogeneous text (same font, size, emphasis, etc) with its precise location. The information is all compressed in a single bit stream, and a UVC program was developed to decode it.

There is no need here to describe the internal format in details. Actually, the advantage of using the UVC is that the details will never be seen by a client; they are fully handled by the UVC program. What is important is that the logical view that is produced be easy to explain. It should contain only data at a logical level, independent of the implementation. The ideal, for example, would be to have the same logical view for any printable document. As we mentioned earlier, this is easy to do for images; it may be somewhat harder but feasible for more complex documents.

In our PDF experiment, we still keep the page image in JPEG. Since both the coordinates and the page image come from the same PDF rendering, they are fully compatible and a viewer can exploit that correlation. For example, it can perform a text search on a given keyword and highlight its occurrence(s) on the displayed page. But keeping page images is costly in storage; they should be compressed in a lossless manner. One solution is to keep the images included in the document in their original format (JPEG, TIFF, GIF, to name a few), and compress the text in a more efficient way, using techniques such as those employed in JBIG2[10]).

Beyond Presentation

This paper focused mainly on the preservation of information that can be printed or displayed. The section on the PDF experiment mentions only briefly the fact that bookmarks were also archived. A bookmark is an example of information that is not really part of the printed output but should be archived nonetheless. The current trend is clearly towards digital documents containing more and more of such “hidden” but essential information.

Process Unrelated to Presentation

A spreadsheet application often uses formulas to compute certain cell values. In some cases, only the results are of interest and the printed table contains all essential information. Other authors may want to keep the semantic information that formulas convey, but only as textual comments. (This is the method that was adopted in a proof of concept conducted in the Netherlands on the archiving of spread sheets.) This is still a very static view of formulas. If the responsible archivist believes that the capability of being able to evaluate the formulas in the future is essential, the problem changes drastically. First, the computation process must be archived; second, the presentation must be re-evaluated since the length of the data elements may change the column widths and/or number of lines. Only a preservation method that preserves processes can do the job.

Marked-Up Documents

Finally, consider documents that are marked up, using a language such as SGML, XML, or HTML, for example. SGML and XML separate the logical mark-up of the information from its presentation. HTML mixes both types of information in the same language. For XML and SGML, a style sheet must be supplied to specify the presentation. While XML and SGML have been or are the subject of intense standardization efforts, style sheets are more complex and less standardized. Since the mark-up provides semantic information, it cannot be replaced by presentation; it must be archived in any case. As far as presentation is concerned, archiving the style sheets and the mechanism to apply it to the text seem to be overkill. Even if the future user should want to reformat the document, it is very unlikely that he would be interested in learning the antique style sheet mechanism and language. Oneapproach is to avoid saving the style sheet, and to preserve the look of the document as discussed above for PDF. If this is not adequate, the style sheet mechanism itself must be defined as a UVC program.

Electronic Publication

Simple publications on CD-Rom’s may be archived simply by archiving the various documents – including musical pieces – individually, together with the metadata stored in the header directory. This is the interface that the CD-Rom player sees. It will be easy to implement such a player in the future. But consider now an educational CD-Rom which contains a complex logic for navigating from one page to the other, or even a video game which is no data, only process. In such cases, emulation of the original code may well be the only solution; it may be a niche but one that cannot be avoided.

The UVC can play a role here, as well. It provides a way to write today an emulator of a current machine in a language that is defined to last “for ever”. The UVC interpreter in the future will be able to run that interpreter; the future client will not need to know anything about the original machine. Emulators have been built successfully in the past but there were built when both the original and new computers were available. This is not the case for the long-term. The UVC approach may be very helpful but the whole approach needs more research.

Conclusion

The problem of digital preservation for the long-term is very challenging. There is no single solution to the problem but several methods have been proposed, depending upon the format of the digital file and the type of functionality required. In some cases, a simple conversion done once at archival time may be sufficient. In other cases, a UVC approach allows data to remain in its original form and simply be converted on demand, at access time, into an easy to understand logical format. Still, a hybrid approach consists of converting the original format into an intermediate format and using a UVC program to decode that format. Finally, a niche exists for applications that require full emulation.

The definition of printable documents is evolving. The essential information of a modern document will be the images and the text in its preferred presentation, but also additional essential information such as comments, bookmarks, links, and above all, logical mark-ups. An ideal archival format would take care of these various types of information in an integrated manner. The future user should then be able to access any subset of the information and to exploit the relationships between the different types of data. The UVC approach however does not advocate the use of one or a necessarily small set of formats since the details of the formats are hidden from the user anyway. Minimizing the number of internal formats decreases the number of UVC programs needed; minimizing the number of logical formats simplifies the task of providing viewers in the future.

References

1. Raymond Lorie, Long-term Preservation of Digital Information, Joint Conference on Digital Libraries, ACM/IEEE, Roanoke, VA, 2001.
2. Raymond Lorie, A Methodology and System to Preserve Digital Data, Joint Conference on Digital Libraries, ACM/IEEE, Portland, OR, 2002.
3. Jeff Rothenberg, Ensuring the Longevity of Digital Documents, Scientific American, (272) 1, 1995.
4. Jeff Rothenberg, Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation, CLIR, 1999.
5. C. W. Brown and B. J. Shepherd: Graphics File Formats, reference and guide, Manning Publication, 1995.
6. Adobe Systems Incorporated: PDF Reference. Second Edition, Adobe Portable Document Format, version 1.3, Addison-Wesley, 2000.
7. Raymond Lorie, The UVC: a Method for Preserving Digital Documents – Proof of concept, IBM/KB Long-term Preservation Study, IBM Netherlands, 2002.
8. Ghostscript, Aladdin Enterprises, 2001.
9. BCL Computers, Inc., http://www.gobcl.com.
10. See www.jpeg.org

Biography

Raymond Lorie is a Research Staff Member at IBM Research. He graduated in 1959 as Ingenieur Electricien-Mecanicien from the University of Brussels (Belgium), joined IBM Belgium in 1960, and IBM Research in 1973. For two decades, he worked on various aspects of the relational database technology; and was a major contributor to System R, a precursor of various relational products. Lorie co-invented the GML mark-up language; he also developed a general method for exploiting context in OCR systems. His involvement in digital preservation started in 1995 when he proposed the UVC preservation method.

Dr. Lorie is an ACM Fellow and co-recipient of the 1988 ACM System Award, for developing the relational technology.