ELECTRONIC PUBLISHING, VOL. 8(4), 343–367 (MARCH 1996)

Portable documents: problems and (partial) solutions

DAVID W. BARRON

Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ
UK

SUMMARY

The paper presents a wide-ranging survey of the issues that arise in producing portable documents, including multimedia and hypermedia documents. It is directed at practitioners, and the approach is therefore pragmatic, based on the current state of the art; the paper does not attempt to provide a comprehensive survey of all previous work on this topic. The nature of an electronic document is discussed, and the various kinds of portability that may be required are defined. Ways in which portability can be achieved in a variety of restricted contexts are presented, including approaches to portability based on international and *de facto* industry standards. The likely success of the competing standards is assessed. Finally the paper addresses the question of whether complete document portability is achievable, or even necessary.

KEY WORDS Portability, SGML, Standards, OpenDoc, ODA, HyTime

1 INTRODUCTION

Corporate information is increasingly viewed, shared, distributed and managed electronically – a recent estimate [1] suggests that eighty percent of all corporate information today exists in digital form. Commercial publishers are moving towards CD-ROM as an alternative to paper, especially for reference and entertainment publications, and the 'digital library' is foreshadowed by the emergence of electronic versions of scientific journals and research papers. This move towards electronic documents highlights the need for portability of such documents over disparate hardware and software platforms, if the maximum benefit is to be obtained from this new technology.

The purpose of this paper is to introduce the issues that arise in producing portable documents. Our approach is pragmatic, and firmly based in the technology available at the present time: we do not attempt to survey all the previous work on portability, much of which has been overtaken by advances in technology and changes in user attitudes. (For example, much of the early work on portability was directed towards the problem of moving documents between word processors with incompatible file formats but all based on a 'typewriter' output model. Solutions to this problem do not encompass 'typeset' documents incorporating graphics, nor are they relevant to the world of hypermedia documents.)

The paper is structured as follows. We first consider the nature of electronic documents in Section 2, providing a working definition of the term 'document' as used in the electronic

CCC 0894–3982/96/040343–25

Received 14 February 1996
Revised 6 November 1996

context and categorizing the different types of document that may be encountered. Then, in Section 3, we offer a wide-ranging discussion of exactly what portability means, and why document portability is important; we identify four largely independent components of portability, define three different forms of portability, consider the issue of visual fidelity to an original, and discuss some particular problems that arise in achieving portability, This part of the paper finishes with a discussion in Section 3.4 of *transportability*, the need to reconstruct internal data structures which have been converted into a stream of bytes for transmission between systems, and the unexpectedly complicated problem of ensuring that such a byte stream is transmitted with 100% fidelity.

Section 4 introduces some of the pragmatic partial solutions to the problem in use at the present time, including approaches based on common interchange formats and platform-independent software. Standards are an important route to portability; we discuss this aproach in Section 5 before going on to discuss relevant *de facto* industry standards (including PostScript, PDF and HTML) in Section 6 and relevant International Standards (especially SGML) in Section 7. Finally, in Section 8 we address the question of whether complete document portability is achievable, or even necessary. The discussion throughout encompasses both 'traditional' documents and hypermedia documents.

The discussion in this paper is limited to the problems of exchanging documents between disparate computer systems: this form of portability can be characterized as 'portability in space'. In the context of electronic archives and digital libraries there is also a need for 'portability in time': this is a much stricter requirement, since the archival form must be independent of the technology, making it possible to re-create documents in the future using hardware and software whose characteristics are completely unknown at the time the archive is created. We explore this aspect of portability in a companion paper [2].

2 DOCUMENTS

2.1 Electronic documents

Before we can discuss portability of documents, we need a precise definition of what we mean by the term 'document', or to be more precise, *electronic document*. The Association of American Publishers (AAP), in its *Reference Manual on Electronic Manuscript Preparation and Markup* [3] defines a document in the following way:

> "A document is an organised collection of smaller pieces of text (such as chapters) and images (such as figures) that are called elements. The elements in a document have a relationship to each other which gives the document a definite organization called document structure"

This definition is too restrictive for our purposes: instead we start from the Oxford English Dictionary definition (one of four) of a document as

> "something written, inscribed, etc., which furnishes evidence or information upon any subject, as a manuscript, title-deed, tomb-stone, coin, picture, etc."

When computers were first used for text processing, documents were composed solely of text, and an electronic document could be defined in similar terms, substituting 'something that can be printed' for 'something written'. A document was the same thing as a file (though text processing by computers pre-dates files: in the earliest systems the document

was a deck of cards or a reel of punched paper tape). Advances in technology, in particular the advent of the laser printer, made it possible to incorporate figures (line art or bit-map images), 'typeset' tabular material, mathematics and chemical formulae into a document. More recently it has become possible to include video sequences, animations and sound clips, creating a new class of electronic documents: as a working definition we define an electronic document as

> "a structured collection of information objects (text, images, video, sound etc.) stored in digital form, which furnishes evidence or information upon any subject."

Current developments (c.f. Sections 2.2, 6.3 and 6.3 below) are leading to a situation where not all the components of a document are necessarily stored within the document: instead the document contains references to objects stored elsewhere. Indeed, as Carr has observed [4], "a document is just a view on a collection of objects". We therefore modify our definition of an electronic document to

> "a structured collection of information objects (text, images, video, sound etc.) *or references to such objects* stored in digital form, which furnishes evidence or information upon any subject."

When it is necessary to distinguish mixed-content documents from text-only documents we describe them as *compound documents*. In what follows, for brevity we use the term 'document' to mean a compound document comprising text and graphics only, i.e. one that can be printed (or viewed on a screen at lower resolution). A document that includes video and/or sound as well as text and graphics, which cannot be printed and must be 'read' on a suitably equipped computer, will be described as a *multimedia document*. A further dimension of distinction is that between documents with a linear structure, and those which include hyper-links that allow non-linear navigation through the document. The traditional definition of hypertext is as a plain text document with links incorporated: many documents nowadays contain graphics and other media, and a multimedia document with hyperlinks is described as a *hypermedia document*. The term *hypertext document* is used only when it is necessary to distinguish pure text documents with links from multimedia documents with links.

2.2 The document as a container

Although an electronic document typically has mixed content, at the level of the operating system in current systems it is still a file — a linear stream of bytes. A document is characterized by the application that can process it, and 'understands' the internal structure of the file: for example, a document produced by Microsoft Word[1] is described as a 'Word document'.

An alternative view decouples the document from the file, treating it as a container which holds a collection of information objects, each with an associated content type e.g. paragraphs (text), images (GIF, JPEG, etc.). This collection of objects may be stored in a file whose internal structure is an encoding of the container structure; alternatively

[1] In the remainder of this paper, 'Microsoft Word' and 'Microsoft Windows' will be abbreviated to 'Word' and 'Windows' respectively

the container structure can be encoded in a file on its own, with pointers to the files that contain the component objects. This view of a document as a container underlies the OpenDoc architecture and, to a lesser extent, Micosoft's OLE architecture (see Section 6.3). The container concept extends naturally to encompass multimedia documents: it is only necessary to add new content types, e.g. video sequences (MPEG), audio (WAV) and so forth. A MIME[2]-encoded multi-part mail message is a simple example of a container document.

World Wide Web (WWW) documents are an interesting variation on the container theme. A WWW document is a hypermedia document which incorporates links (pointers) to other documents that can be located anywhere on the Internet, and these documents can in turn contain links to further documents. These links are only activated at the explicit request of the user: we can say that the WWW document represents a 'virtual document' which is the collection of information obtained by activating all the links, though the user can only see one component of the document at a time. (Computer scientists will recognize this as an instance of lazy evaluation.)

2.3 Meta-documents

An increasingly important form of document is the *meta-document*: a structured collection of information from which instances of documents and other resources can be derived. Examples include:

- The Oxford English Dictionary which exists as a database from which are derived various printed editions (Shorter, Concise, Pocket etc.), as well as the CD-ROM version
- Critical editions of a literary text, where a single source 'document' contains all the variations, and can be printed out using different variants as the base text

3 PORTABILITY

3.1 Components of portability

It is the nature of an electronic document that it can be transmitted in digital form over a network, or recorded digitally on a physical medium (disk, CD-ROM) and then loaded onto another computer system. If the hardware and software platform of the target system is identical to that of the originating system the document can evidently be processed and rendered on either system with equal facility. Portability is the attribute that makes processing and/or rendering possible when the target system differs from the originating system in hardware platform, software platform, or both.

It is important to observe that there are four separate components of a document to be considered in achieving portability:

- content
- logical structure
- layout structure
- presentation

[2] Multipurpose Internet Mail Extensions

A major step towards real portability is to keep the four components strictly separated: one reason that proprietary systems do not produce portable documents is that these four components are not clearly separated in the digital representation of the document. Rather, they are irretrievably bound up in the internal structure of the file holding the document. Often, information that would aid portability is thrown away. For example, consider what happens when a user inserts an image in GIF format into a Word document. The image is converted into a Windows metafile (a sequence of instructions for the Windows GDI, the graphics rendering engine) and although it is identified as a picture in the internal representation, it is now portable only to systems that 'understand' Windows metafiles, and not to non-Windows applications that "understand" GIF files.

3.1.1 Content and logical structure

Content is the blocks of text, images etc. that make up the document. The logical structure defines the relationships between them, e.g. a paper in this journal has head matter, body and tail matter in that order; head matter consists of title, author details, summary and keywords again in order; the body consists of paragraphs and figures assembled into a hierarchy of sections and sub-sections, and so on. A hypertext/hypermedia document also has a *navigational structure* — the collection of hyperlinks — which can be regarded as part of the logical structure; the logical structure of a hypermedia document may also include temporal relations, e.g. the length of time for which a video clip is to run, or synchronization requirements between video and audio clips.

It is self-evident that to achieve any kind of portability we must be able to preserve content: except for very limited forms of portability we must also preserve the logical structure.

3.1.2 Layout structure and presentation

Layout structure and presentation are important components of printable documents, but have less importance for hypermedia documents. In a printable document, layout structure determines the positioning of blocks of content on the printed page, providing visual clues to the logical structure, whilst the presentation determines the actual typefaces and sizes employed, use of bold and italic type and other decorations. Note particularly that by 'layout' we mean the positioning of a block of content *in relation to neighbouring blocks*: other matters that might be regarded as part of the layout such as indentation, justification etc. are treated as part of the presentation. Hypermedia documents may have some layout and presentation structure, but these aspects are normally determined partly by user preference (e.g. window positions) and partly by constraints imposed by the underlying hardware (e.g. the size of the window in which video clips can be shown).

The extent to which we retain layout structure and presentation information determines the degree of visual fidelity that will be achieved in a printable/viewable document. Since, in general, portability is inversely related to the degree of viusual fidelity required, it is desirable wherever possible to relax this requirement. This is not generally an option for documents with non-text content: degradation of graphics is in many cases unacceptable — engineering drawings, for example, must be reproduced with absolute fidelity to the original — whilst the meaning of tabular matter, mathematical equations and chemical structure diagrams is conveyed in good measure by their two-dimensional layout, and

comprehension will be hindered if visuial fidelity to the original is not preserved.

Absolute visual fidelity of textual matter is less important, unless it is required to maintain a corporate image or to preserve the 'look and feel' of a printed work when it is converted to electronic form. This is sometimes a requirement in the legal world, where quotations from a standard work of reference are used to back up a case, and the appearance of the printed page carries authority. Publishers converting journals to electronic format also tend to require good visual fidelity in order to maintain their 'brand image' and, as in the legal case, to maintain the authority of the journal. Visual fidelity is also used as a form of authentication; digital documents can be changed with ease, and the visual fidelity to a familiar paper-based original provides a (possibly superficial) assurance that the electronic document is an accurate reproduction of the original. Finally, absolute visual fidelity is required in the unlikely event that the meaning of the text depends on its layout. Two examples of this can be adduced.

1. In the English legal system an interpretation of a statute by a judge in a higher court becomes part of the Common Law, binding on lower courts. On rare occasions a judge has reached an interpretation of an ambiguous point on the basis of the physical layout of the printed statute, and when converting old statutes to electronic form the precise layout is preserved as a precaution.
2. In Lewis Carrol's classic, *Alice's Adventures in Wonderland* [5], the dormouse's tale (a poem) is set as a sequence of very short lines with a left indentation which varies in a sinuous manner. The point-size of the setting reduces progressively from 14-pt on the first line to 2-pt at the end, so the visual appearance of the poem reminds the reader of a rodent's tail. The dormouse's *tale* is printed to look like the dormouse's *tail*: the joke would be lost if the setting were not preserved[3].

3.2 Forms of portability

There are three distinct forms of portability. The first, portability for printing and viewing, requires that it possible to render the document on the target machine and, in the case of a hypertext/hypermedia document, to follow the links. The second form of portability requires that it is possible to edit the content and alter the formatting to a greater or lesser extent: it may also extend to creating new links to or from a hypertext/hypermedia document. Where this form of portability is provided it usually appears in combination with the first form, but it can appear on its own. The third form of portability is a variation on the second: it allows printing and viewing, and also allows certain types of limited processing, as explained below.

3.2.1 Portability for printing or viewing

This is the most common variety of portability, and applies when the document is to be read or viewed by a human recipient, and is not to be edited or re-formatted on the target system. We can distinguish two categories of users who require this form of portability.

- Professional publishers who are moving to electronic delivery of books and journals in CD-ROM or other electronic formats (e.g. the World Wide Web).

[3] I am indebted to David Brailsford for reminding me of this example.

- Geographically distributed organizations[4] which have a requirement to move commercial or technical information between different sites.

In the case of the professional publishers we have a one-to-many distribution requirement. The second category is in general a more *ad hoc* many-to-many distribution requirement – frequently it arises from the need to print arbitrary documents at remote sites rather than delivering paper by conventional means. An example is the Physics research community with its electronic pre-print archive (see Section 4.5 below). Another example is furnished by the CALS[5] intiative [6] in the USA. This requires all defence contractors to submit tenders in electronic form, and further requires successful bidders to provide documentation for the resulting weapon system as electronic documents.

For documents made up of text and graphics (line art and bit-mapped images) there are three possible levels of portability:

1. The document is rendered with absolute visual fidelity to the original.
2. The document is rendered with approximate visual fidelity to the original.
3. The document as rendered has the same content and abstract structure as the original but does not necessarily preserve any of the presentation or layout aspects.

Absolute visual fidelity means that a printed version of the document produced on the source and target systems will be identical in the sense that a photocopy and its original are identical. (This may not always be achievable. For example, if a document which includes a 24-bit colour image is sent to a system that only supports 8-bit colour, absolute fidelity is clearly impossible, in the same way that absolute fidelity is lost if a colour document is copied on an ordinary photocopier.) Approximate visual fidelity relaxes this requirement: typefaces in the two versions may not be identical, though the font metrics will usually be the same, ensuring that line breaks are preserved. Page breaks may not be preserved if the two systems use differing paper sizes. At the third level of portability the sequencing of the sections and paragraphs will be maintained, but the page layout may be completely different, e.g. single column in the original and double column on the target machine.

Similar considerations apply to multimedia documents, although presentation or layout of such documents is largely determined by the viewing software and the preferences of the user.

3.2.2 Portability for document processing

The need for an alternative kind of portability arises in a distributed workgroup environment where members of a geographically separated group have to work together on a project, perhaps sending a document to a remote site where some processing is carried out prior to returning the document. The weakest requirement is that processing and formatting can be carried out on the target machine even though the platform differs from the originating system. A stronger requirement is to be able to control the degree to which processing and formatting can take place. For example, the Open Document Architecture (ODA), which is discussed in more detail in Section 7.1, allows the author to specify whether arbitrary

[4] An organization in this sense may be a business enterprise, an academic institution, or a heterogenous community with a common interest.

[5] The acronym CALS originally stood for 'Computer Aided Logistical Support'. By the time the project was launched in 1988 it had changed to 'Computer-aided Acquisition and Logistic Support', and more recently it has changed yet again, to stand for 'Continuous Acquisition and Lifetime Suppport'.

changes in content, layout and presentation can be made, or whether the document content can only be edited and/or re-formatted within constraints specified by the originator. In particular, it is possible to specify that although editing and re-formatting are allowed, no change can be made to presentation (thus allowing a house style to be preserved).

A common version of limited processing is digital proofing. This consists of adding comments and annotations (an electronic version of the 'Post-It note') to a document that originated at another site: the document with its annotations is then returned to the original author for revision in the light of the comments. Another example of limited processing is the addition of a personal set of hyperlinks to a document. Adobe Acrobat Exchange is a good example of a system that provides both kinds of limited processing capability.

3.3 Problems of portability

The major source of problems is of course the potential difference between the platform on which a document is originated and that to which it is ported: if the platforms are identical portability is assured. This is only the case, though, if the platforms are totally identical — the same hardware, the same software in the same version and at the same revision level, the same keyboard layout, the same character coding convention and the same collection of fonts. Substantial problems can arise when the platform is almost identical, but not quite.

3.3.1 Problems with fonts

Availability of fonts is a major problem affecting electronic document delivery: if the target machine does not have a font that is used in a document it is not possible to render those parts of the document that use the missing font. (At least, not comprehensibly: typically a PostScript printer will default to Courier, though the rendering software is assuming the metrics of the missing font. This causes words or parts of words to overlap, and lines to be the wrong length.) In theory portability can be achieved by embedding all the fonts used into the document. However, this falls foul of licensing restrictions in most cases, and the embedded fonts cause a gross inflation in file size, leading to problems if transmission bandwidth is restricted. Methods for alleviating the font problem are discussed in Section 4.4.

3.3.2 Problems with national character-sets

Any document that uses characters outside the standard English (US) alphabet poses portability problems: this is a matter of particular concern for a pan-European enterprise. The problems arise from differences in the character coding conventions adopted by the systems involved. For example, in character text using the ISO 646 code [7], the byte code 0x23 represents a pound (currency) sign for a Windows system with a UK keyboard (or a system using the UK, French, Italian or Spanish interpretation of ISO 646), but represents a 'hash' sign to a US version of Windows (or a system using any of the other 14 national interpretations of ISO 646). (Futher problems of portability arise in this particular case from the US habit of using the name 'pound' for the hash character. This arises from an old custom of appending '#' to figures denoting a weight in pounds.) This is but one example of the effect of national variations of the ISO 646 character set. There are 12 characters subject to national variations: for example, code 0x7D corresponds to a closing brace (}) in the UK variant, e-grave (e) in the French variant and u-umlaut (u) in the German variant.

Thus a document produced in France using this convention will be garbled if ported to a German system using ISO 646, and editing the document using a German keyboard will be possible, but unintuitive.

Most systems attempt to solve the problem of accented letters and diacritical marks by using an 8-bit character set. Unfortunately there are many incompatible sets. DOS-based systems use a series of 'code pages' containing characters for particular groups of languages. Windows uses 'Windows-ANSI' which corresponds closely to the ISO8859-1 (Latin 1) character set, incorporating glyphs for most European languages. Thus a Windows document is reasonably portable to another Windows system: the text will display correctly even if it employs a combination of languages, though editing such a document will be tedious, since it will involve entering numerical codes for characters that are outside the set for which the keyboard is intended. (And a port of such a document from Word for Windows to Word for Macintosh will be disastrous.) This problem is addressed in detail by Reese [8].

In principle this problem will disappear when all systems support multi-byte character code sets such as ISO10646 [9] and Unicode[6] [10] [11] (except for the problems of transmitting 8-bit characters dicussed in Section 3.4 below). It will be some time before such codes are generally supported — at the time of writing only Windows NT provides full support for Unicode — and at present the only reliable way of achieving portability with non-English character sets is to use SGML: as described in Section 7.2 SGML provides a uniform way of referencing a large range of characters, e.g. `&eacute;` for e. This is the method used in the Text Encoding Initiative Guidelines [12] but is not an option in many cases.

3.3.3 Problems with hypermedia links

Links in hypermedia (and hypertext) documents present a major obstacle to portability. In many systems the links are embedded in documents in a system-specific format, and the linking capability of systems differs. Thus in a text document some systems specify the target of a link as a byte offset within the document, whereas others may only allow links to the start of a structural object such as a section or sub-section. Clearly a document produced by a system of the first kind is not completely portable to a system of the second kind. The so-called 'Dexter model' of hypermedia [13] originated from early work on hypermedia interchange formats, but it remains an abstract model, not followed in practical implementations.

Open hypermedia systems, of which Microcosm [14] is the leading example, store links in a separate database called a 'linkbase'. This is a text file, and is therefore readily portable. Although it would in principle be possible to translate links from the linkbase into the native format of another system and then embed them in a document, in practice Microcosm documents are only portable to another system running Microcosm.

3.4 Portability and transportability

An underlying requirement for portability is that documents are *transportable*. A compound document will be stored internally in a more or less complicated data structure which has

[6] ISO646 defines a 4-byte canonical form UCS-4, but also specifies a two-byte subset UCS-2, the 'Basic Multilingual Plane', which is essentially the same as Unicode, which was defined by a consortium of vendors.

to be reduced to a linear byte-stream before transmission over a network or recording on magnetic media. Transportability is the property of being able to effect this reduction and subsequently re-constitute the data structure unchanged on the target system. Note carefully the difference between portability and transportability. Transportability ensures accurate reconstruction at the byte level, but the meaning ascribed to identical bytes may differ between systems, as we have seen in Section 3.3.2 above.

A general approach to transportability requires agreement on a non-proprietary scheme for linearisation of the document data structure; fortunately there are well-established standards for this — the International Standards ASN.1 and BER. ASN.1 (Abstract Syntax Notation 1) [15] is a language in which data and data structures can be defined in a manner independent of particular machine representations, and can therefore be used to describe a structured document file in machine-independent terms. BER (Basic Encoding Rules) [16] is a *transfer syntax* that defines an unambiguous translation between structures defined in ASN.1 and a linear byte-stream.

The requirement that the linear byte stream that is produced in this way (or by some proprietary encoding scheme) can be faithfully transmitted over a network appears deceptively simple: in practice, pitfalls abound. For example, some e-mail systems pad lines to a fixed length with spaces: others strip off trailing white space at the end of a line. Even a simple text file is not transportable between a Unix system and a Windows system because of the different conventions used to mark the end of a line – carriage-return and linefeed in Windows (DOS) files, just a linefeed in Unix files. Thus the layout will be lost when the file is transfered. (Transfers in the other direction are usually possible, since most text-processing utilities under Unix will ignore the extra carriage return characters. However, this cannot be guaranteed.)

The main cause of trouble, however, is that the transmission medium may not be 'eight-bit clean', assuming that data is encoded in a 7-bit code such as ASCII, and using the eighth bit for its own purposes. File transfer protocols usually include provision for specifying eight-bit transmission (e.g. the 'set binary' or 'set image' command of the Internet file transfer protocol), but an increasingly common requirement is to transfer portable documents by electronic mail (e-mail), and for this purpose the file must be encoded as a stream of 7-bit ASCII or ISO 646[7] characters, since e-mail transmission cannot be guaranteed to be 8-bit clean. There are four well-established ways of achieving this:

Quoted-printable encoding This encoding leaves printable 7-bit (ASCII or ISO646) characters unchanged, but represents the control characters and any 8-bit characters by an '=' followed by a two-digit hexadecimal representation of the byte. Thus the ASCII form-feed character is encoded as '=0C'. (Catch-22: A genuine '=' appearing in the text must be encoded as if it were an 8-bit character, i.e. '=3D'.)

Binhex This is the standard encoding method for Apple Macintosh systems, and treats each byte by two hexadecimal digits represented by the ASCII characters 0–9 and A–F.

uuencoding The uuencode program (and its companion uudecode) originated in the Unix world as a way of encoding binary files for e-mail transmission, and is now widely used in the PC world. It splits each group of three bytes into four six-bit units which

[7] For most practical purposes, ASCII is identical to the US interpretation of ISO 646 (X3.4-1968).

are mapped onto the first 64 printable characters in the ASCII (ISO646) set (codes 0x20–0x5F).

Base-64 This encoding employs the same six-bit groupings as uuencode, but differs from uuencode format in the control characters used.

If the sender and recipient are both using MIME-enabled mail systems, the document can be sent as an attachment and necessary conversions will be performed transparently. Users of less spohisticated mail systems have to perform the encoding and decoding of the document explicitly.

This is not the end of the story on e-mail transmission, however. An e-mail message may pass through many intermediate systems before reaching its destination, and problems are likely to arise if a message is routed via an IBM mainframe that uses EBCDIC coding. EBCDIC exists in at least six mutually incompatible versions, so a translation from ASCII to EBCDIC and back again does not necessarily preserve the original ASCII characters. This does not affect binhex files (the upper-case letters and digits map into themselves on all translations), but it can be a problem for uuencoded files and text files. Characters which are not guaranteed to survive 'blind' transmission include square brackets and braces ([ ] { }), currency signs (£ $) and a collection of miscellaneous characters (# @ | \), and to achieve guaranteed transportability these should be avoided. Quoted-printable encoding can be extended to cover these characters: alternatively the SGML 'entity' syntax (see Section 7.2 below) can be used for text files. Before transmission an opening brace, for example, can be replaced by the sequence `&ob;`, and at the receiving end the special sequence can be recognised and the opening brace restored. (Both parties must agree on what `&ob;` means, e.g. by agreeing to use the entity definitions for Latin-1 characters defined as part of the SGML Standard.) This technique is not suited to base-64 encoded or uuencoded files, since the sequence '`&<char><char>;`' is likely to occur within the encoded file[8] and for guaranteed transportability binhex should be preferred.

A simple one-to-one correspondence between a document and a file cannot be guaranteed: a document may be spread over multiple files, and as future systems based on container architectures are developed this will become the norm. For such documents, transportability requires a way of specifying how the components are packed into a data stream and unpacked after transmission. For example, the SGML Document Interchange Format (SDIF) [17] defines a way in which this can be done for SGML documents.

At this point we should note that whilst meta-documents have to be transportable, the concept of portability applies only to the document instances that are derived from them.

4 AIDS TO PORTABILITY

4.1 Agreement on the platform

The challenge of portability is to be able to render and/or process a document on disparate hardware and software platforms. The problem disappears if management *fiat* can ensure uniformity of platforms. (It must however be total uniformity as defined in Section 3.3 above.) In the business world this may well be possible: in many large enterprises hardware and software are purchased on a company-wide basis, and are likely to be identical over large parts of the organization. However, smaller businesses often show evidence of the

[8] As an experiment we took the dvi file produced by LaTeX for this paper as a 'typical' binary file: the special sequence occurs in 17% of the lines of the uuencoded version.

'candy store' effect in their purchases, and indeed there may be legitimate reasons for different parts of a large company to use incompatible platforms. This may be no barrier to portability since the market is dominated by applications — WordPerfect, Word, PageMaker etc. — which exist in compatible form on a wide variety of hardware/software platforms. Similarly, for more complex technical documentation requirements, systems like InterLeaf and FrameMaker run both on Unix and (powerful) PC platforms. 'Agreement' may be imposed by Government, as in the CALS initiative (see Section 3.2.1 above): tenders for weapons systems are only accepted if presented in a strictly defined portable electronic form. As a counter to this, in some organizations standardization by management decree is just not possible: for example, in the academic world the choice of word processor or DTP package arouses passions and disagreements rivalled only by those between competing religious sects or the supporters of rival football teams.

Software suppliers are encouraging this approach by aggressive selling of integrated suites incuding Microsoft Office, Lotus SmartSuite and Corel Office7[9]. In the case of hypermedia documents, agreement on platform is the only practicable way of achieving portability at the present time.

4.2 Conversion programs

A simple approach to portability is to use conversion programs to map one proprietary format onto another. These usually take the form of import and export filters: for example, Microsoft Word will convert an input file from WordPerfect format, and will write a Word file in WordPerfect format, and similar conversions are provided by WordPerfect. However, the conversions are rarely completely successful, and almost always need some manual tweaking due to the way undocumented features have been mapped. In particular, it is not uncommon to find quite bizarre font substitutions taking place.

Achieving portability for multimedia documents is likely to depend heavily on the use of conversion programs. Examples include reducing the colour depth of an image from 24 bits to 8 bits, converting a colour image to grey-scale, conversion between different graphics formats (e.g. GIF to JPEG). Many products are available to perform this kind of conversion.

4.3 Common interchange formats

A variant on conversion filters is the use of a platform-independent notation for describing a formatted document containing text and graphics. One such notation in common use is Microsoft's Rich Text Format (RTF). The specification of RTF [18] [19] describes it as 'a method of encoding formatted text and graphics for easy transfer between applications'. Portability is aided by the fact that it is a non-binary format, using ANSI, PC-8, Macintosh or IBM PC character sets. It can describe formatted text that includes graphics, but only if these take the form of Windows or OS/2 metafiles, Windows bitmaps or Macintosh QuickDraw files. Many word processors provide import and export filters for RTF, though it has to be said that as an interchange medium it suffers from the same problems as conversion programs, especially as regards font substitution.

[9] Previously marketed as Wordperfect PerfectOffice.

4.4 Viewers, readers and page-turners

If the inter-working requirement is restricted to being able to render a document on a number of platforms, a popular solution is the use of a *viewer*. A viewer is simply a program that will display or print a document without having to run the application that created the document. For example, Microsoft WordView allows a Windows user to view/print a Word document without having Word installed.

Simple viewers are restricted to processing files produced by their 'parent' application. An increasingly popular approach to electronic distribution of portable documents is to combine the idea of a viewer with a common interchange format so that it is possible to view documents generated on different platforms by diverse applications: the name *reader* is commonly used for such viewers. A reader typically provides additional functionality, e.g.

- Facilities for navigating within a document, enabling the user to read the document rather as one would read a book (e.g. by providing links from a table of contents to the relevant page).
- Thumbnail views of a collection of pages.
- 'Post-It Note' annotations.
- Hypertext linking within and between documents.

Readers with this kind of enhanced functionality are sometimes called page-turners: available systems include Adobe Acrobat, Common Ground and Envoy. Of these, Acrobat shows signs of becoming the dominant system: many publishers employ it for electronic publication of academic journals, and it is widely used in the US by government agencies, the military and major industrial companies as a digital storage format. The recent announcement of Acrobat Exchange 3.0, formerly known as Acrobat Amber, which integrates Acrobat with the Netscape Web browser suggests that the Adobe Portable Document Format (PDF) will rapidly establish itself as a *de facto* standard alongside PostScript.

A particular reader has associated with it one or more *writers*: these may be free-standing conversion programs (for example, Acrobat Distiller) that translate documents into the common interchange format, but more commonly they are operating system specific programs that masquerade as a printer driver, and appear in the 'print' menu of word-processing applications as another print device. When such a virtual printer is selected to 'print' a document, it generates a file in the interchange format that can later be interpreted by the associated reader.

The usefulness of readers is enhanced by the fact that they include mechanisms to cope with documents that use fonts which are not available on the target system. Acrobat uses Adobe's 'Multiple Master' font technology [20]. A multiple master font incorporates several sets of outlines so that intermediate versions can be generated along the axes of weight, width and style, as well as the size axis characteristic of outline fonts. It requires that the metrics of the fonts used be embedded in the document, or available via the Adobe Type Manager database of font widths, which holds data for all fonts in the Adobe Type Library. If a required font is not available to the reader, the font metrics are used to generate an approximation from a multiple master font. Although not identical to the missing font, this will have the same metrics and a similar weight, so that the appearance of the printed page will be acceptable: in particular, line breaks and justification will be preserved. This technique produces acceptable visual fidelity for typical body faces, though

for obvious reasons it will not cope with decorative or display faces: these are handled by embedding font outlines in the document. Common Ground originally used the technique of embedding bit-maps of all the glyphs used in a document at 'native' size and at a smaller size for generating 'thumbnails', but now uses Bitstream's 'TrueDoc' technology, as does Envoy. TrueDoc recodes the font(s) in a very compact representation, including only the characters that actually occur in the document: this technique is claimed to avoid any copyright problems that might arise from embedding fonts, though this claim has yet to be tested in the courts.

Readers may be free-standing items of software, e.g. the Acrobat reader, or may be 'bundled' with a document: for example, the Common Ground reader can be embedded in the document. CD-ROM documents routinely include the viewer software (possibly for multiple platforms) on the CD.

4.5 Using machine-independent software

Another way to solve the portability problem is to use software that is machine independent. An outstanding example of this approach is Donald Knuth's TEX [21] system for typesetting. Written entirely in a high-level language, this runs on any machine that has a Pascal or C compiler, and generates a device-independent output, with printer drivers available for almost any imaginable printer. TEX, often front-ended by Lamport's LATEX [22], has been widely adopted in the academic world as a means of exchanging research papers. A spectacular example is the electronic archive of Physics research pre-prints and papers, the e-Print archive [23] held at the Los Alamos National Laboratory, which is used by large numbers of physicists world-wide. (Access rates in excess of 10,000 per day are reported). TEX is very good at handling mathematics, but lacks features for art work: the archive is therefore LATEX based, with diagrams in PostScript. A problem that has to be addressed is that TEX documents may reference files of specially defined macros, and LATEX documents may use style files that are not available on a user's local system. When a paper is submitted to the archive the author provides a header that specifies any macro package or special style files used in the paper, together with an ftp address for retrieving these if they are not held in the collection of macro packages and style files that forms part of the archive.

5 STANDARDS AND AGREEMENTS

The very notion of portability implies some form of agreement between the originator of the document and its subsequent user. They may agree to use the same hardware and software, rendering the problem almost trivial, or they may choose to use applications that are designed to inter-work. Standards are just widely accepted and well documented agreements: for example ISO Latin 1 [24] defines a unique correspondence between a set of glyphs and a set of integer codes, and applications that agree to use this standard coding are guaranteed to inter-work at that level (provided that they do not require to use codes which have 'national variants' in the Standard, and that the operating system supports the use of such a character set).

Most standards fall into one of two classes: National and International Standards developed by governmental and inter-governmental bodies, and "industry standards" which are a codification of widely accepted practice at the grass-roots level. A prime example of this latter category is PostScript, which is almost universally adopted as the page descrip-

tion language for laser printers and image setters. Industry standards are by their nature "bottom-up": they arise from the spread of a particular way of doing something, initially implemented by one manufacturer. Traditionally, International Standards are developed "top-down", starting from an abstract requirement to develop a standard for something. In the computing world the most successful standards (with one or two honourable exceptions) are formalisations of existing bottom-up industry standards, possibly with some tidying up. Top down standards in this field are rarely very successful: an obvious example is ODA, which after some 15 years of development still has no applications in the marketplace[10]. (SGML is an evident exception to this generalization.)

A third class of standards are those which arise from the activities of an individual or a small group, who develop software that is made freely available at no charge. Some such software is so widely used that it amounts to a *de facto* standard: an apposite example is Knuth's TEX system.

6 PORTABILITY BY USE OF INDUSTRY STANDARDS

6.1 PostScript and PDF

If the portability requirement is restricted to printing and/or viewing, and the documents are restricted to text and graphics only (i.e. no video or audio) the increasing availability of PostScript printers can make PostScript a useful medium for portable electronic delivery. This is particularly the case in communities such as the Computer Science research community, or the technical program development community, where PostScript printers are ubiquitous. For example, Microsoft and many other companies make technical documentation available in PostScript form from ftp and Web servers. Provided the document creators restrict themselves to the 35 fonts that come with all PostScript printers, absolute visual fidelity is guaranteed.

The PostScript-based Portable Document Format (PDF), the interchange format on which Adobe Acrobat is based, provides an enhanced level of portability. PDF [25] is a file format that can represent a compound document containing text and graphics in a manner independent of the application software, operating system and hardware used to create it. PDF is a sophisticated interchange format: a PDF file includes font metrics, it can include compressed versions of text and images, and it incorporates a cross-reference table to the pages and other major structures in the document. The cross references in the file allow the reader software to provide page-turning facilities, and the font metrics provide a partial solution to the problem that arises from non-availability of fonts as described in Section 4.4. As noted earlier, Acrobat Exchange provides limited hypertext linking facilities and a simple searching capability.

6.2 HTML

In theory, World Wide Web (WWW) documents are highly portable thanks to the use of a common mark-up language (HTML)[11]. This portability is gained at a price, since the

[10] The 'Compound Document Architecture' (CDA) developed by the Digital Equipment Corporation was based on the ODA document model.However, although CDA was promoted as an open architecture, it remained a proprietary DEC product.

[11] Strictly speaking, HTML is an application of SGML, being defined by an SGML 'Document Type Definition'. However, hardly any documents conform strictly to HTML as defined in this way, and it is probably best to

document author loses almost all control over the presentation of the document, which is at present almost entirely determined by the browser software. (Work currently in progress on 'Web style sheets' [26] may change this situation in the near future.) The author of an HTML document can label the logical components of a document, e.g. a heading, but the presentation will depend on the browser and the local options set by the user (if any). Indeed, even the size of the window in which the document will be displayed is unknown to the author. Moreover, the portability of HTML documents depends on authors restricting themselves to the standard agreed version of HTML. Netscape browsers implement an extended version of HTML (the "Netscape extensions") with the result that it is possible to author documents that render superbly with Netscape, but come out as a total mess when rendered by other browsers: the same is true of Microsoft's Internet Explorer. It is in principle possible to get round the problem by providing pages in duplicate forms, one version using proprietary extensions and the other using a 'vanilla' version of HTML, since the HTTP protocol provides for the browser to send an identification string as a basis for content negotiation between browser and server. This can be used to select the appropriate version of the pages, but in practice little if any use is made of this capability.

6.3 OLE and OpenDoc

OLE and OpenDoc are two competing architectures for compound documents. In this section we briefly describe and compare the architectures, and assess their potential as vehicles for portability.

6.3.1 OLE

'Object Linking and Embedding', from which the acronym OLE was originally derived, originated as a technology developed by Microsoft (and others) to implement a container architecture for compound documents. The first version, OLE 1, provided fairly basic facilities, and was later enhanced as OLE 2. However, OLE has since developed into a technology for developing object-based systems that is central to the Microsoft view of the future of desktop computing, and is rapidly becoming the foundation on which all Microsoft operating environments are built.

OLE is built on the 'Component Object Model' (COM)[12] which provides the 'low-level plumbing' that enables transparent communication between components through standard interfaces. It encompasses a number of technologies of which only 'OLE Documents' and 'OLE Persistent Storage' are relevant to the current discussion. (Other major components include OLE Controls (OCXs), now called 'ActiveX controls', OLE Automation and OLE drag-and-drop.) These components provide:

1. a container-based document architecture based on COM, in which documents or parts of documents can be encapsulated as objects which can then be embedded in other document objects, and
2. a storage format which allows structured compound document objects to be stored as a flat file.

treat it as special-purpose mark-up language.

[12] Confusingly, the acronym COM is also sometimes used to refer to the DEC/Microsoft 'Common Object Model'), a specification defined by the two companies to facilitate inter-working between their disparate object systems (ObjectBroker and OLE).

An application that is OLE-aware can act as a *server*, a *container*, or both. A server application can define part of a document (a paragraph, a graphic image, a range of cells in a spreadsheet etc.) as an object that can be exported. A container application running on the same machine can import such objects, either physically embedding a copy in the document, or by using a pointer to the object's parent document. Linking ensures that if the server application changes the content of an exported object, the changes are reflected in the container document: it also conserves space, which may be important if the object is a large image file, for example. However, linking requires that the original file does not move, and if this cannot be guaranteed, embedding must be used. An attraction of embedding is that it allows "in-place editing": for example, an Excel spreadsheet object embedded in a Word document can be processed by Excel without leaving Word. (If the object is linked it must be edited in a separate window.). At present, OLE only works between servers and containers running on the same computer. However, the architecture was designed with distributed operation in mind, and the recent announcement of the Distributed Component Object Model (DCOM) opens the way to this.

Although OLE is a major component of Microsoft systems, OLE Documents technology is not restricted to Microsoft applications: any developer of software to run under Windows can choose to make an application OLE-aware, and able to act as a container, a server, or both.

6.3.2 OpenDoc

OpenDoc is a 'vendor-neutral' container architecture developed under the auspices of Component Integration Laboratories, a non-profit consortium of companies including Apple, IBM, Novell, Oracle and others; the actual development of the core product is carried out by Apple and IBM. (Novell were involved at this level, but withdrew.) It is currently available for the Macintosh (and will be an integral part of the Macintosh operating system in its next release), and a beta-test version for Windows is available: it is stated policy of CIL that OpenDoc and related products "will be available on most major operating systems including MacOS, MVS, OS/2, OS/400, UNIX, VM/CMS, and Windows".

OpenDoc introduces a major paradigm shift: unlike OLE, which is built round the concept of container and server applications, in OpenDoc a document is not associated with a single parent application but is composed of smaller blocks of content called 'parts': these can have a variety of of content models, and have an associated with them a collection of part-editors and part-viewers, which perform the manipulations on the document. Thus a JPEG-encoded image part would have an associated viewer (and possibly editor) specialised to JPEG images; a text part might have multiple editors and viewers, using the conventions of different word processors to avoid the need for re-training. An OpenDoc document has a recursive structure, i.e. a part can contain other parts, thus providing an arbitrary degree of embedding in a hierarchic manner. The architecture provides a clean separation between the container document and the processes that operate on it, and the use of a collection of part editors and part viewers is a welcome move away from huge monolithic applications towards 'component-ware'. Like OLE, OpenDoc is built on an underlying object model, in this case the IBM 'System Object Model' (SOM), and its extension the 'Distributed System Object Model' (DSOM). Again like OLE, the architecture has two major components.

1. A SOM/DSOM-based container architecture, in which parts are instances of object

classes, and the part editors and viewers are the methods associated with the object class.
2. An underlying storage model in which an arbitrarily complex structured collection of parts can be stored in a flat file.

Whilst OLE is offered as a vendor-neutral and platform-independent architecture, the designers recognise the entrenched position of Microsoft, and hence OLE, in the marketplace. They have therefore made provision for interworking between the two architecures. Wrappers are provided to make an OpenDoc container appear to be an OLE container, and to wrap OpenDoc parts so that can be embeded in an OLE container application, with the appropriate part editors playing the role of the OLE server.

6.3.3 Comparison of OLE and OpenDoc

Superficially, OLE and OpenDoc sound very similar, but there are major differences.

- In OLE, in-place editing of an embedded object is done by the monolithic server aplication to which it 'belongs'. Editing and rendering of an OpenDoc document is the responsibility of the collection of 'part editors' and 'part viewers' associated with each different type of part.
- The underlying storage model of OpenDoc, called Bento [27], has many useful features not present in the OLE equivalent. In particular it allows multiple versions of a given part to exist simultaneously. This can be used as a form of version control, alternatively a single version of a part may be stored in two or more representations. For example, a text part might exist both in a form suited to manipulation by a program and in a form adapted to efficient editing from the keyboard.
- Distribution is just in prospect for OLE, whereas DSOM is an established technology which makes it possible for components of a document, and/or the part editors and viewers, to be srored at various locations in a distributed computer system.

In addition to these differences, there are two more subtle differences that arise from differences between the COM and SOM object models. We give here a simplified description for the benefit of non-technical readers.

- SOM/DSOM is a classic object model which supports the concept of "inheritance": COM does not support the concept. This means that it is possible to develop an OpenDoc part editor that inherits most of the properties (and re-uses most of the code) of an existing part editor, but is specialised in some particular respect. For example, if it is required to manipulate text with complex tables, a part editor can be produced which inherits most of its behaviour from a 'parent' editor, but replaces the table editing capability with its own more powerful version.
- COM is a static-linking model: SOM provides binary-level sub-classing and (effectively) dynamic linking. What this means in practice is that it is easy to 'plug in' a modified version of an OpenDoc part editor, or to develop part editors using different programming languages. For example, in the scenario outlined above, the original part editor might be written in C++, and the extension for dealing with tables more effectively could be written in COBOL[13]. COM does not confer this sort of capability.

[13] Unlikely, but possible.

The upshot is that OpenDoc provides a more flexible way of developing component software. However, whether that is a contribution to portability remains to be seen.

6.3.4 Implications for portability

An OLE-embedded object has two associated pieces of data: the *presentation data* that is needed to display the object, and the *native data* that is required to edit it. An OLE document can therefore be displayed on any system running the container application, and thus has the same limited portability for viewing and rendering as any 'plain' document produced by the same proprietary application. If the document is to be processed by the receiving system, however, that machine must have available all the server applications for the embedded objects, e.g. if a Word document with an embedded Excel object is moved to another system, that system must have both Word and Excel installed in order to edit the spreadsheet component.

The above discussion is restricted to OLE documents that employ embedding only. In a linked object the presentation data is present as before, but the native data is replaced by a link to the server application. OLE-linked documents are thus inherently non-portable, since the links encode specific file path-names in the file system of the originating system.

OpenDoc will not make a major contribution to document portability unless it becomes so widely used as to become a *de facto* standard. In the short term its use is likely to decrease portability, since it will be necessary to replicate the whole collection of part viewers (and part editors if the document is to be processed) at the destination site. In principle this could be obviated by using the distributed system model with the parts being collected as required over the network: however, whether such a distributed system architecture will catch on is by no means a foregone conclusion, and in any case the architecture is only the first step – the problems of charging for remote use of a collection of distributed components is horrendous, and may be insuperable.

On the other hand, the OLE and Bento storage formats provide an attractive route to transportability for compound documents. The Bento storage model is particularly well suited to the representation of multimedia documents, and if it were to be adopted as the 'standard' storage format for such documents would make a great contribution to the portability of multimedia documents. This need not imply that the whole OpenDoc architecture has to be adopted, though that does seem to be a promising way forward for portable multimedia.

7 PORTABILITY BY USE OF INTERNATIONAL STANDARDS

7.1 ODA

ODA, the Open Document Architecture, is the current incarnation of one of the earliest attempts to establish a framework for portable documents. At the beginning of the 1980's, before the advent of the PC, word processing was performed on dedicated proprietary systems, and business was plagued by the incompatibility between them. Each had its own proprietary file formats, and document interchange was impossible. In 1981 the European Computer Manufacturers' Association (ECMA) embarked on a project to establish an 'Office Document Architecture' as a compound document interchange structure that that would overcome these problems of incompatibility. The first ODA standard [28] was

published by ECMA in 1985, and was adopted as an International Standard [29] in 1988. It was also adopted as a Recommendation by CCITT (now renamed ITU-T). ODA in its current form supports compound documents made up of text and graphics (metafile and bitmap) only. ECMA and ITU-T have changed the name to "Open Document Architecture" to reflect an intention to encompass multimedia documents: the same change will appear in the next issue of the ISO standard.

The designers of ODA were particularly innovative. They recognised the concept of a compound document and the difference between portability for printing and portability for processing. ODA defines three forms of document transfer:

1. Formatted form. The document is reproduced exactly as intended by the originator (i.e. with absolute visual fidelity).
2. Processable form. The document content can be edited and/or re-formatted within constraints specified by the originator.
3. Formatted-processable form. Arbitrary changes in content, layout and presentation can be made.

Despite its apparent advantages, however, there are few if any ODA applications on the market. This is probably because the gestation time of International Standards is incompatible with the rapid change in computer technology. ODA was conceived at a time when word processing was based on special-purpose hardware and software, and the printing technology was essentially that of a typewriter. By the time the standard was completed the world had changed: word-processing was done on desktop computers and output was pseudo-typeset text produced by a laser printer. Although work is still in progress on a new version of the standard, by the time it arrives the world will have moved on.

7.2 SGML and related standards

The Standard Generalized Markup Language (SGML) [30] is an International Standard which defines a meta-language for the description of a class of markup languages which in turn define a document structure. The user constructs a *Document Type Definition* (DTD) which defines

- a collection of *tags* that label the structural components in a document, e.g. section, subsection. Tags can have associated *attributes* which make it possible to provide information for use in down-stream processing of a document. For example, the tag that labels a piece of artwork can be defined to have as an attribute a measurement of its depth, so that the appropriate amount of vertical space can be reserved by a formatting program.
- the relationship between the tags, and the consequential constraints e.g. sub-sections can only appear within an enclosing section. In this respect the DTD is a formal grammar describing document structure.
- *entities* with associated *entity references*. These can be used to provide links to external files, as a shorthand notation for long text strings, and – most importantly – as a way of entering text characters that do not appear on the keyboard (c.f. Section 3.3.2 above).

A program called an SGML parser replaces entity references by the associated entity and uses the DTD to verify that the particular document instance conforms to the structure.

Recalling the four components of portability from Section 3.1 — content, logical structure, layout structure and presentation — SGML can clearly be seen as a completely portable way of defining the logical structure of a compound document. Each component has a tag to describe it: the tags for non-text components will have attributes describing the encoding (e.g. GIF, JPEG for images) and the location of the file containing the actual content. By the use of entities SGML also provides a completely portable way of transmitting text content, solving the problems of mixed-language text discussed in Section 4.1. A major example of SGML in this role is found in the *Guidelines for Electronic Text Encoding and Interchange* produced by the Text Encoding Initiative [12]. By using entities for the characters that may be garbled during network transmission (see Section 3.4) SGML also provides a solution to the potential problems of 'blind' transmission.

However, SGML says nothing about layout or presentation, which are the province of two further standards, the Document Style Semantics and Specification Language (DSSSL) [31] and the Standard Page Description Language (SPDL) [32]. These Standards have only recently been ratified, and as yet there are no commercial applications that make use of them. Currently available SGML applications act as a front-end to an established document processing system (e.g. Interleaf, Frame) and the resulting documents are only as portable as those produced by the back-end system on its own.

In theory, when all the standards are agreed there will be a basis for portable documents: whether it will displace proprietary solutions on a wide scale is a matter for conjecture.

7.3 HyTime

The Hypermedia/Time-Based Structuring Language, HyTime for short [33,34,35], provides a standardized way of defining the scheduling and interrelationships of the time-dependent components of a hypermedia document (audio and video), and provides an alternative to proprietary 'scripting languages' such as Apple's QuickTime. for describing time-dependent material. In addition, it provides a way of defining positions within a document in a system-independent manner, and thus opens up the possibility of defining hypermedia links in a transparent and portable fashion, overcoming the problems outlined in Section 3.3.3. HyTime provides a set of abstract facilities that can be built into an SGML-based document architecture, and its use therefore pre-supposes the use of SGML to describe the logical structure of the document. Given that hypermedia documents do not in general have a strong requirement for absolute fidelity of layout and presentation when moved to a different system (c.f. Section 3.1 above), the arguments advanced against SGML in the previous section do not apply, and SGML/HyTime promises to be a good combination for obtaining portability of hypermedia documents.

7.4 MHEG

MHEG [36] is a Draft International Standard for the coding of multimedia and hypermedia information that can be used by and interchanged between applications. It defines a container architecture in which media objects can be stored using appropriate external standards (e.g. SGML, ASN.1, JPEG, MPEG, etc.), together with instructions for their presentation and behaviour. The intention is to provide a practical interchange standard for hypermedia documents, and may provide an alternative to HyTime/SGML as a way of achieving portability, though there is as yet little practical experience of its use. However, it is

important to note that it is restricted to *presentation*, and is not really suitable for interchange of documents with revisable components. This is because the display and control semantics are embedded in the MHEG standard, whereas in HyTime these aspects are devolved to the controlling application.

8 THE REQUIREMENT FOR UNIVERSAL PORTABILITY

8.1 An heretical question

It is apparent from the foregoing dicsussion that the universally portable document has much in common with the Holy Grail (or the Snark [37]), greatly sought after but never to be found. OpenDoc is hailed as the route to portability, but has only recently come to the market. SGML and its related family of standards offer the most promising route to a universal solution: unfortunately, though SGML-based products are available at the present time these only address part of the problem — portability of content and structure. The standards further down the line that would cover portability of layout and presentation have not yet come into general use, having only recently ceased to be the subject of debate and argument. There is a further problem in that OpenDoc and SGML-based solutions are all-or-nothing solutions, requiring users to abandon their present technology to make way for a superior replacement. If we were starting electronic publishing from scratch, OpenDoc or SGML might be the way to go: unfortunately, we have to live in the real world. Given the aggressive marketing in that real world of software suites, technical documentation systems and products like Adobe Acrobat, and the immense investment in this existing technology by users, it is unlikely that there will be an immediate rush to OpenDoc or SGML-based systems when they eventually arrive, except possibly for multimedia documents.

However, we have seen in the earlier sections that there exist a number of partial solutions to the portability problem that meet the needs of particular categories of users, and we are led to ask the heretical question “do we need total portability?”. Engineers have the concept of a solution that is “good enough”: are we nearing that point in document interchange? In the following sections we explore “good enough” solutions for various kinds of document.

8.2 Printable documents in final form

We first consider documents that are composed of text and graphics, which will ultimately be printed, or rendered on a screen as a simulacrum of the printed page. This category comprises the most common form of electronic document at the present time, and for such documents the most commmon portability requirement is to make possible electronic delivery of documents to customers/clients with a diverse array of ‘reading’ devices over a network or on CD-ROM[14]. For the majority of such documents this requirement is adequately met by page turners such as Acrobat and Envoy. Readers are available free of charge, and if the material is published on CD-ROM readers for a variety of platforms can be bundled on the CD, making the document highly portable. (Computer scientists will recognise this as an instance of a *closure*.) Indeed, it is likely that before long we shall see PDF (and possibly other) readers bundled into the software portfolio when machines are

[14] The CALS initiative provides an interesting inversion of this model, requiring portability so that clients with a diverse array of authoring devices can send documents to single destination platform.

sold. If a document of this kind is destined just to be printed on receipt, PostScript is in many cases a perfectly adequate vehicle. (So is a fax machine.) However, using Acrobat or similar systems confers added value in the form of annotation, hyperlinks and searching.

These systems do not provide a universal solution: there are some categories of documents for which other routes to portability must be sought. One example is a technical document with a large component of mathematics and/or other symbolism in its content. Portability of such documents is usually achieved by agreement: the Physics e-Print archive shows that TEX provides an acceptable solution for a particular community, and in the technical documentation industry Interleaf and FrameMaker are *de-facto* standards. Another example is literary texts: the work of the Text Encoding Initiative has shown that SGML provides an effective way of achieving portability of such documents. SGML is in fact an efective way of achieving portability for any documents where layout and presentation is not a concern, and is the preferred solution for such documents, since it is almost 'technology proof', requiring only the ability to handle a basic character set such as ASCII or ISO 646.

8.3 Printable documents in processable form

Considering now documents that have to be exchanged in processable form to support cooperative working, we first observe that in the main this takes place *within* an enterprise or a group with closely matched interests (e.g. computer science researchers, scholars working on textual criticism), not *between* enterprises or groups with diverse and disparate interests. Most enterprises nowadays (the academic world excepted) adopt a company-standard software portfolio. We are no longer in the world that led to the ODA activity: at that time it was the norm for different parts of an enterprise (particularly a multi-national) to have incompatible word-processors but nowadays the company is likely to have standardized on Word, Word Pro[15] or WordPerfect, and the portability problem is solved (except for possible problems with character sets as discussed in Section 3.3.2). Thus for this category of documents a common (or compatible) software platform is 'good enough'. For the increasingly important field of digital proofing, Acrobat and similar systems certainly provide a 'good enough' solution.

8.4 Hypermedia documents

The situation is much less clear-cut for hypermedia documents, and the only effective form of portability at present is agreement to use a particular proprietary authoring and/or delivery vehicle. Hypermedia documents distributed on CD-ROM can have the viewing software bundled on the CD, and are then portable to any target machine that has the requisite multimedia hardware (sound card etc.). As yet there is no satisfactory way to exchange multimedia documents over a network except for the limited facilities offered by the World Wide Web, but this is not at present a major requirement in most enterprises. Here we have the kind of green-field scenario in which OpenDoc or SGML/HyTime may succeed when they are eventually available on the market.

[15] The current (confusing) name for Lotus Ami Pro

9 CONCLUSION

We should accept that for the forseeable future there are going to be multiple approaches to portability, and concentrate our efforts on achieving "good enough" solutions. The universally portable document, turns out to be not the Holy Grail of document processing, but a chimera.

> "**chimera** *n* An unreal creature of the imagination, a mere wild fancy; an unfounded conception. (The ordinary modern use.)" [16]

REFERENCES

1. *TrueDoc Font Technology Overview*, Bitstream Inc, 215 First Street, Cambridge MA 02142, USA, 1995.
2. D.W. Barron, 'Reliable archiving of electronic documents'. (To appear).
3. *Reference Manual on Electronic Manuscript Preparation and Markup*, Association of American Publishers, 2005 Massachusets Avenue, NW. Washington DC 20036 USA, 1989.
4. L.A. Carr, *Structure in Text and Hypertext*, Ph.D. dissertation, University of Southampton, UK, 1995.
5. Lewis Carroll, *Alice's Adventures in Wonderland*, Macmillan, London, 1865.
6. Joan M Smith, *An Introdution to CALS*, Technology Appraisals Ltd, 1990. ISBN 1-871802-04-0.
7. *International Standard ISO 646: 7-bit Coded Character Set for Information Interchange*, International Standards Organisation, 1983.
8. A.R Reese, 'E-mail for more than plain English', *AXIS*, **2**(2), 18–23, (June 1995).
9. *International Standard ISO/IEC 10646: Universal Multiple-Octet Coded Character Set (UCS)*, International Standards Organisation, 1993.
10. Unicode Inc., *The Unicode Standard Volume 1*, Addison Wesley, Reading MA USA, 1991.
11. Unicode Inc., *The Unicode Standard Volume 2*, Addison Wesley, Reading MA USA, 1992.
12. *Guidelines for electronic text encoding and interchange*, eds., C.M Sperberg-McQueen and L. Burnard, University of Illinois, Chicago, USA, 1994.
13. F.G. Halasz and Schwartz M, 'The dexter hypertext reference model', in *Proceedings of the NIST Hypertext Standardization Workshop*, Gaithersurg, MD USA, (January 1990).
14. H. Davis, W. Hall, I. Heath, and R. Wilkins, 'Towards an integrated information environment with open hypermedia systems', in *Proceedings of the Fourth ACM Conference on Hypertext*, pp. 181–190, Milan, Italy, (1992).
15. *International Standard ISO 8824: Specification of Abstract Syntax Notation 1 (ASN.1)*, International Standards Organisation, 1989.
16. *International Standard ISO 8825: Specification of Basic Encoding Rules for Abstract Syntax Notation 1 (ASN.1)*, International Standards Organisation, 1989.
17. *International Standard ISO 9069: SGML Document Interchange Format (SDIF)*, International Standards Organisation, 1988.
18. *Microsoft WORD Technical Reference*, Microsoft Press, Redmond, WA USA, 1990.
19. *Application Note GC0165: Rich Text Format (RTF) Specification*, Microsoft Product Support Services, Redmond WA USA, 1994.
20. *The PostScript Font Handbook*, Addison-Wesley, Reading MA USA, 1992.
21. Knuth D E, *The TEXBook*, Addison-Wesley, Reading MA USA, 1984.
22. Lamport L, LATEX, *a Document Preparation System*, Addison-Wesley, Reading MA USA, second edition, 1994.
23. The archive can be accessed at http://xxx.lanl.gov.
24. *International Standard ISO 8859-1: 8-bit Single-Byte Coded Graphic Character Sets – Part 1: Latin Alphabet No. 1*, International Standards Organisation, 1987.
25. Adobe Systems Incorporated, *Portable Document Format Reference Manual*, Addison-Wesley, Reading MA USA, 1993.

[16] Oxford English Dictionary definition

26. Information about the current status of work on Web style sheets can be found at http://www.w3.org/pub/WWW/Style/.
27. J. Harris and I. Ruben, *Bento Specification*, Apple Computer Inc., Cupertino, CA, 1993.
28. *ECMA-101: Open Document Architecture (ODA) and Interchange Format*, European Computer Manufacturers' Association, Geneva, second edition, 1988.
29. *International Standard ISO 8613: Office Document Architecture (ODA) Part 1–8)*, International Standards Organisation, 1988.
30. *International Standard ISO 8879: Standard Generalised Markup Language (SGML)*, International Standards Organisation, 1986.
31. *International Standard ISO/IEC 10179 Document Style Semantics and Specification Language*, International Standards Organisation, 1996.
32. *International Standard ISO/IEC 10180 Standard Page Description Language*, International Standards Organisation, 1995.
33. *ISO/IEC Standard 10744: Hypermedia/Time-based Structuring Language (HyTime)*, International Standards Organisation, 1992.
34. S. Newcomb, N. Kipp, and V. Newcomb, 'The Hytime hypermedia/time-based document structuring language', *Communications of the ACM*, **34**(11), 67–83, (1991).
35. L.A. Carr, D.W. Barron, H.C. Davis, and W. Hall, 'Why use Hytime?', *Electronic Publishing: Origination, Dissemination and Design*, **7**(3), 163–178, (1994).
36. *Draft International Standard ISO/IEC DIS 13522-1 (MHEG Part 1)*, International Standards Organisation, 1995.
37. Lewis Carroll, *The Hunting of the Snark*, Macmillan, London, 1876.