Separable Hyperstructure and Delayed Link Binding

David F. Brailsford

University of Nottingham
School of Computer Science and Information Technology
NOTTINGHAM NG7 2RD, UK

Abstract: As the amount of material on the World Wide Web continues to grow, users are discovering that the Web's embedded, hard-coded, links are difficult to maintain and update. Hyperlinks need a degree of abstraction in the way they are specified together with a sound underlying document structure and the property of separability from the documents they are linking. The case is made by studying the advantages of program/data separation in computer system architectures and also by re-examining some selected hypermedia systems that have already implemented separability. The prospects for introducing more abstract links into future versions of HTML and PDF, via emerging standards such as XPath, XPointer XLink and URN, are briefly discussed.

Background

Research on electronic hypertext systems stretches back for more than 30 years to Engelbart's [Engelbart 1968] and Nelson's [Nelson 1987a] astonishingly far-sighted work. A body of hypertext transcends normal textual documents not only by allowing user-specific annotations to be added to the corpus, but also by enabling a set of intra- and inter-document hyperlinks to weave documents together into an associative `web'. Indeed, today's World Wide Web can be regarded as a reification -- albeit flawed and full of compromises -- of what these pioneers had in mind.

Over the years the idea of linking has evolved from being a purely text-related activity into the wider world of `hypermedia' and its philosophy of linking, for example, text strings to associated sound bites, icons to video clips and indeed `anything to anything'. In the spirit of this new generality the term `document' is used, in what follows, to denote not just textual matter but any form of material that can act as the source or destination of hyperlinks.

It turns out that among all the decisions to be made about the way hyperlinks are implemented, one issue is particularly worthy of analysis and debate: are the links to be tightly bound, as an inseparable part of the material to which they apply, or does the implementation allow a degree of link separability so that links can be held externally in some form of *link base*? On the face of it this seems to be a simple issue of whether a separate link base can help in maintaining the integrity of large numbers of links, but there is far more to it than that. The property of hyperlink separability leads inexorably into the need for a model of structure in the underlying documents; the degree of document object granularity in that model; the way in which link sources and destinations are anchored, together with the type properties of links and their traversal semantics.

FINAL DRAFT of survey paper which appeared in ACM "Computing Surveys" 31(4es) December 1999.

In a survey of this brevity we cannot do justice to all the various approaches that have been adopted over the years. Instead we shall comment on just a few systems and how these have influenced the approach taken in emerging standards such as XPath, XLink and XPointer. The underlying theme is that the integrity of a large collection of shared documents can best be guaranteed if the documents themselves are kept `pure' in some sense. More specifically, users of the documents can supply their own `added value' of personalized annotations and hyperlinks but these should be kept separate from the document itself and applied flexibly to it, as and when needed. We must move away from inadequate document architectures which compromise a document's integrity by requiring that users' personal links and annotations be rigidly embedded into the document itself.

The arguments to be made for purity and separability are not new; they simply revisit, with some fascinating new twists, the evolution of computer system architectures to enable programs to be shared safely, at run time, by multiple users.

Re-entrant programs as a paradigm for separable hyperstructure

In later sections we discuss how the separability of hyperstructure is a desirable starting point for maintaining the integrity of a shared, hyperlinked, set of documents. When the links and other annotations are overlaid onto the document at execution time it becomes vital to have a strong underlying document structure to enable these features to be located at the correct points and to function correctly. We can now draw some useful analogies between the emergence of document architectures which have these desirable properties and the evolution of computer architectures which allow multiple users, with personalized data sets to share one copy of an application.

The earliest stored program computers of the 1940s and 1950s made no attempt to distinguish between those bit patterns in memory that corresponded to the program, as opposed to those that were data. Indeed, such was the cost of memory that saving a few memory locations by writing self-modifying code or by `executing' a stored constant (if it happened to correspond to a useful instruction!) were seen as the very hallmark of powerful programming. But the appalling problems of debugging and maintaining such programs were just one factor in forcing a radical reappraisal. A more potent factor still was the realisation that binding any single user's data, tightly and intimately, into the very program that processed it, was utterly inimical to the idea of sharing that same program code, among many users, in the newly-emerging generation of time-sharing machines. The pioneering work on paging for the ATLAS computer in 1964 and the subsequent adoption of the ideas in the IBM 360/370 series and in the MULTICS operating system showed the virtues of a rigid separation of code and data, enforced by hardware mechanisms with operating system help. By contrast with a tightly-bound, *impure*, mix of program code and data in first-generation computers these new systems enabled programs to be *pure*, or *re-entrant*, with the same code being used simultaneously by multiple different data sets. The purity of the code was reinforced by having hardware-enforced `read only' tags attached by the operating system to the code pages of users' programs. Over the years these ideas were steadily refined, leading to powerful multi-user operating systems such as UNIX and the development of advanced chip architectures such as the Motorola 680x0 and SPARC series.

A host of new abstractions was needed to shield from users the fact that ultimate execution of a program took place at fixed, absolute, memory addresses. This process of abstraction started with symbolic procedure names and private namespaces within programming languages. It then progressed via dynamic (shareable) linked libraries and remote procedure calls (these latter guaranteeing unique names over a network) to encompass the idea of uniquely named objects that could persist for as long as they were needed.

It is depressing (but perhaps inevitable) that when computers become an affordable mass-market commodity -- via the IBM PC and its successor clones -- they were forced by reasons of cost, chip availability and a host of non-technical factors to adopt a hardware and software architecture that was initially shackled to single-user operation and offered an architecture little better than a first-generation computer of the early 1950s. Only now, after 25 years of development have modern Pentium chips and Windows NT enabled the PC to attain the sophistication of shareable code and shareable dynamic linked libraries available to users of MULTICS in 1972.

Interestingly this story of advance, retrenchment and regrouping applies with equal force to hypertextual architectures. Systems which supported link separation were available more than 15 years ago but today's World Wide Web has a document architecture, in HTML, which is an uneasy combination of structural and layout features and where every document has to have its hyperlinks hard-coded within itself.

Hypertext architectures

A body of hypertext reverses the roles of program and data as set out in the previous section. The corpus of documents corresponds to `the data'; the various sets of interpreted hyperstructure that can be overlaid on this data correspond to various possible `programs'. In what follows we shall concentrate exclusively on hyperlinks but it is worth remarking that the argument for separability applies with equal force to other items of hyperstructure such as bookmarks, electronic `sticky notes' and indeed to anything that can be regarded as an annotation of the document which personalises it for a given user or group of users.

So, if different users apply different sets of hyperstructure to the same corpus of basic material then we have a situation of multiple programs being applied to a single set of data. But, this time, multi-user access, and a sharing the common material, requires that the data, rather than the program be kept `pure'.

The possibility of separated linkbases was foreseen more than 15 years ago (another example of pioneering foresight by Engelbart [Engelbart 1984] and Nelson [Nelson 1988] and was firmly established as part of hypertext practice by the time of the Intermedia [Garrett 1986] system. More recent systems such as Microcosm [Davis 1992]] and Hyperwave [Maurer 1996] have enthusiastically developed the concept. Quite apart from all other considerations this approach allows multiple linkbases from different sources to be applied to the same corpus of material and a host of link typing information to be maintained. The integrity of the linkbase is easy to maintain and the links can be kept in an abstract form independent of any given hypertext system. Moreover, these abstract links can not only be *interpreted* by the specialist hypertext system to which they apply, but they can also be *compiled* into any of the target formats such as HTML and PDF that demand embedded `hard links' [Brailsford 1998].

To embed or not to embed?

Although we have just mentioned some potent arguments for keeping links in separate linkbases it is not by any means a one-sided case. It is equally possible to find arguments against such a separation, and for keeping the links embedded within the material to which they relate. For example, embedded links ensure that a document can be edited, and moved as a self-contained entity. However, if this document is not only the source of outgoing links but also the target of an incoming link action, then moving the document's position in a file hierarchy, or altering any of its destination nodes that act as anchors for incoming links, would very likely cause the linking action to fail.

To focus our ideas a little more on these separability issues we can begin by distinguishing three degrees of link separation as outlined by Davis [Davis 1995]

(1) Embed all node and link information within the document
(2) Embed tags to define source and destination nodes for "persistent selections" but store the links themselves externally.
(3) Store both the link, and pointers to potential nodes, externally

The next step is to examine the degree to which the links, whether internal or external, are `hard-coded' or pre-computed. Even for externally held links we can distinguish two extremes:

(a) The links are between pre-existing fixed nodes within documents whose identity and location are also known in advance.
(b) The source and destination link nodes, together with the names and locations of the linked documents are specified procedurally; the link becomes fully resolved by computing the results of these procedures at link binding time.

All kinds of intermediate variations exist between the extremes of (a) and (b) listed above. Even if link nodes are not visible within a document they can often be held, internally or externally, in pre-computed form, as byte offsets into the file, or as a bounding box of a `view' on a given page. At the other extreme the link manager might compute at linking time the names and locations of source and destination documents which may, in turn, be in a format that the link manager cannot directly interpret (e.g. video data or data in a format uniques to some particular application). In such cases the link action will need to invoke interfacing software, at either the source or destination, which presents potential link nodes in a form that the link manager can understand.

Given the possibilities we have just outlined it is disappointing indeed to find that just as the IBM PC reverted to a previous generation of hardware and system design so also have two of today's important de facto standards in electronic documents (HTML and Acrobat/PDF) reverted to an approach of hard-coded embedded links. Within HTML the links are tagged with the document via the now-familiar:

```
<A HREF="mydestination">myanchor</A>
```

construction. To be fair, this form of tagging does at least enable the links to be textually demarcated for possible future extraction into some external link-base. Adobe's PDF is far worse; it is a page-based format (with an imaging model derived from PostScript Level 2) and is essentially dominated by considerations of `appearance' rather than `structure'. Although it implements intra- and inter-document links (including an option for Web traversal) it does so via a hidden data structure within the file that does not surround, or even explicitly point at, any textual anchor for the link. Rather, if the words `hello world' are to be highlighted as a source anchor of a hyperlink on a PDF page, then the page number and bounding box of that phrase and those of the intended link destination, have to be calculated and stored in an appropriate pdfmark structure.

Reverting to the analogy of computer architectures we can only hope that these two important document formats do not take 25 years to evolve a form of link that is both abstract and separable. Two recent developments give cause for hope. The first is the work under way within the World Wide Web consortium to develop a schema for link and pointer specifications [DeRose 1999]. The second is the decision by Adobe Systems Inc. to make hidden structure tags (including a tag for hyperlinks) be part of PDF from PDF 1.3 (and Acrobat 4) onwards [Adobe 1999].

Anchoring the links -- the need for document structure

As we have now seen, an abiding problem with any scheme that separates links from their potential anchors is how to locate the required anchors, at source and destination, and how to associate the required links with them at link-binding time. All of this has to be implemented in a system where the underlying material is to be kept pure and shareable (and which may have been made `read only', via file system or CD-ROM hardware constraints, in order to achieve these very properties). Indeed, one has only to read of the various subterfuges forced upon a hypertext system such as Microcosm [Hall 1996] to realise how difficult it can be, in separated linkbase systems, to overlay the links onto underlying material which has little or no inherent structure. At the crudest level it is sometimes possible to locate anchors via fixed byte offsets into a file, or via bounding boxes of character strings and image material in graphically-based systems such as Acrobat. Both of these are inflexible hit-and-miss methods which encounter great difficulties when files are edited and would-be anchors move around within the file.

To locate link anchors successfully requires material with a comprehensive set of hidden structural tags that can act as node points for attaching hyperstructural features. Logically, therefore, a scheme such as the Standardised Generalised Markup Language (SGML) [Goldfarb 1990] -- which is a metasyntax for defining document tagsets -- would seem a good candidate for imposing structure within hyperdocuments. In SGML one writes a document type definition (DTD) which defines the valid tags to be used in a collection of documents and which sets out all the properties of those tags, including an implicit specification of how they combine hierarchically to impose a tree structure on a document. (The analogy, in programming terms, would the use of BNF metasyntax to define the legal keywords, and potential program structures, for all the valid programs that could be written in a language such as C.) And just as a compiler for a computer language builds up the implicit tree structure of the program within itself, so also does an SGML parser build up and validate the implicit tree structure defined by the tags within a document.

Although SGML was a major landmark for structured documents it failed to gain widespread usage, largely because of the need to have the DTD present, as well as the tagged document itself, before parsing could begin. Added to this was a multitude of problems caused by the decision to allow SGML tags to be optionally omitted if the defined tag syntax in the DTD would allow allow the parser to unambiguously insert the missing tags. A programming analogy can illustrate very starkly the problems that this causes. Imagine a language such as Pascal which imposes its nested structure via `begin` and `end` keywords. Imagine furthermore that the language designers allowed `end` keywords to be omitted under certain circumstances, where the compiler, in principle, could infer that such a tag should be present. To compound this parsing nightmare let us add a requirement that the compiler must take great care to distinguish between correct keyword omission (by experts) and incorrect keyword omission (by beginners making mistakes).

For all the reasons we have just set out, full SGML was unsuited as the foundation for a mass-market hypertext system. But interestingly the HTML that underlies the World Wide Web is indeed based on SGML notation though it is important to understand that, unlike SGML, it does not operate at the metasyntactic level -- it is simply a fixed set of tags for marking up documents to be interpreted by Web browsers. This `fixed' tagset has, in reality, proved to be all too incompatibly extensible (by competing browser vendors wanting to add extra features) and, in the early days at least, there was no well-defined DTD for HTML. Add to this the fact that different vendors' versions of HTML allow differing subsets of end tags to be omitted (and yet the absence of a DTD leads to ambiguities about where these missing tags should be inserted) and there is little wonder that well over 90% of HTML documents on the Web are in some sense `illegal'.

HTML is an uneasy mix of tags that are vaguely structural, with many others that are concerned solely with the *appearance* of the Web page. But despite all these weaknesses it has been enormously successful in establishing the principle that a structural notation is the way ahead for electronic documents. It has also introduced a base-level capability for embedded hypertext links (via the `<A>` tag). However, now that a huge body of documentation is available on the Web it has evident that HTML is too weak, architecturally, to support a robust, separable, hyperlinking scheme, and yet adopting SGML as the metasyntactic framework for future Web tagsets could lead to enormous parsing problems. To address this dilemma a team of experts came together in 1998 to propose an Extensible Markup Language (XML) [Bray 1998] which is still a metasyntactic framework (as opposed to a fixed tagset) and yet is also a proper subset of SGML. Many of the troublesome features of SGML have been excluded (not least the optional omission of tags, which is not allowed in XML). The benefits are that a moderately lightweight parser can do some form of consistency checking on a document even if the DTD is not available and the guaranteed presence of a full set of tags lays down a secure framework for link anchors. The tree structures of XML-tagged documents can be manipulated via a Document Object Model (DOM). Any document explicitly tagged in XML notation, and conforming to the DOM, immediately provides a wide range of elements, IDs and attributes that act as location guides to enable the linking software to determine, from an external specification of a link within a linkbase, which elements (or parts thereof) are to form the sources and destinations of a particular hyperlink.

Within a given XML-based tag-set there could indeed be tags identified by names such as `<LINK>`, or `<NODE>`, which denote anchors for the document authors' own links. However, the existence of other structure elements delimiting tables, diagrams, headings etc. gives a framework for arbitrary, per-user, hyperstructure to be attached. To achieve this three extra proposals have been made to the World Wide Web consortium, called XPath, XPointer and XLink [DeRose 1999]. Broadly speaking, XPath provides a mechanism for identifying a particular node or set of nodes in the document's structure tree as a starting point for a link anchor; XPointer elaborates what XPath can achieve by allowing `selections' to be made which transcend node boundaries (for example, a document might be tagged up with every sentence separately identified between `<SENTENCE>` and `</SENTENCE>` markers and yet a user wants to create a link `hot spot' that spans the end of one sentence and the beginning of the next). Finally, XLink specifies the syntax and semantics for embedded and external links including options for an `extended out-of-line' pointer which offers the prospect of multiway links stored in separated linkbases.

For documents that are not innately conformant to XML/DOM structure there is a possibility of imposing an external structure on them via the use of metadata, and this leads, in turn, to the notion of externalising the whole of

a document's structured markup (as opposed to just the link specifications) by means of `standoff markup' [McKelvie 1998].

Resolving links -- the need for persistent document names

Although the introduction of a DOM together with XPath, XPointer and XLink will greatly facilitate the application of external linkbases to corpora of documents, the burden of maintaining links in present-day HTML and PDF stems not only from the fact that the links are embedded within inadequately structured documents, but also because the links themselves point at *where* a target document is stored rather than *what* it is. In other words documents need to be `called by name' rather than being `called by location' and these persistent names should be used as a matter of course in abstract hyperlink specifications.

The resource naming problem we have just identified has been recognised for some time and is being addressed by initiatives such as the Universal Resource Name (URN) [Mealling 1999] and CNRI Handle [Handle 1999] proposals. This is not to say that the adoption of name resolution protocols would banish for ever the horror of broken links and 404 not found messages, but there is every prospect that such a service could automatically track the latest incarnation of a named target document or, at the very least, some metadata that describes it. The problems of scale in having the World Wide Web be able to locate documents by name should not be underestimated but initiatives such as the Digital Object Identifier (DOI), which is a form of URN being proposed and tested by a consortium of international publishers, shows very clearly the e-commerce and e-publishing benefits to be gained from an agreed persistent naming scheme.

Conclusions

The key point to be grasped is the desirability of link structures being *separable* rather than, necessarily, being forcibly *separated*. In principle at least the embedding of links should be just a convenience, to be utilised from time to time if there is a need to `keep everything together' in one file; there should be nothing in the link specifications that prevents extraction and separation in some other different situation. In other words, even if link markup is embedded within the material to which it refers (and there is a strong case that links which an author wants to be part and parcel of a document should be treated in this special way), there should always be a clear distinction (very much along the lines first envisaged in the Dexter Hypertext Reference Model [Halasz 1990] between the data structures for link nchors and link storage, as against those for document presentation.

In this respect having a widely-adopted standard for document structuring is of enormous help and it seems that we can expect ever-increasing pressure to structure documents in XML-approved ways, starting, perhaps, with the adoption of XHTML [Pemberton 1999] (a cleaned up version of HTML designed to be XML conformant). The adoption of the XML metasyntax ensures that a single parser can cope with all sorts of `added value' associated with a document, ranging from links specified via XLink to metadata specified via RDF [Lassila 1999] Other *de facto* document standards such as Adobe's PDF, which does not currently use XML notation for its new range of embedded structure tags, may soon be forced to move in the XML direction, if only to ensure that such documents can inter-work reasonably seamlessly with a new generation of XML-compliant Web documents.

Nevertheless the adoption of XML constitutes no more than a desirable pre-requisite for what we want to achieve. The realisation of separable and abstract hyperstructure, for digital documents on the World Wide Web, requires solid progress on such diverse issues as late binding of externally specified links onto appropriately structured documents, and a robust resolution scheme that enables us to access digital resources on the Web by name rather than by location.

References

[Adobe 1999] Adobe Systems Incorporated, Portable Document Format Reference Manual version 1.3, April 1999.

[Brailsford 1998] David F. Brailsford, Steve G. Probets, Les Carr, and Wendy Hall, "Dynamic Link Inclusion in Online PDF Journals" in Proceedings 7th International Conference on Electronic Publishing (EP98), pp. 550≠562 , Springer Verlag, 1998.

[Bray 1998] Tim Bray, Jean Paoli, and C. Michael Sperberg-McQueen (editors). Extensible Markup Language (XML) 1.0, Cambridge, Massachusets: World Wide Web Consortium, [Online: http://www.w3.org/TR/REC-xml], February 1998.

[Davis 1992] Hugh Davis, Wendy Hall, Ian Heath, Gary J. Hill, and Rob J. Wilkins. "Towards an Integrated Information Environment with Open Hypermedia Systems" in Proceedings of the ACM Conference on Hypertext (ECHT '92), Milano, Italy, 181-190, [Online: http://acm.org/pubs/citations/proceedings/hypertext/168466/p181-davis/], December 1992.

[Davis 1995] Hugh C. Davis. "To Embed or Not to Embed..." in Communications of the ACM (CACM), 38(8), 108-109, August 1995.

[DeRose 1999] Steven J. DeRose. "XML Linking" in ACM Computing Surveys, Symposium on Hypertext and Hypermedia, 1999.

[Engelbart 1968] Douglas C. Engelbart and William K. English, "A Research Center for Augmenting Human Intellect" in Proceedings of the Fall Joint Computer Conference (IFJC), 33, Arlington, VA, pages 395-410, 1968.

[Engelbart 1984] Douglas C. Engelbart. "Authorship Provisions in Augment" in Intellectual Leverage: The Driving Technologies, 465≠472, Compcon84, IEEE Computer Society, 1984.

[Garrett 1986] Nancy L. Garrett, Karen E. Smith, and Norman K. Meyrowitz, "Intermedia: Issues, Strategies and Tactics in the Design of a Hypermedia Document System" in Proceedings of ACM CSCW '86, Austin, TX, 163-174, December 1986.

[Goldfarb 1990] Charles F. Goldfarb. The SGML Handbook, Oxford University Press, 1990.

[Halasz 1990] Frank G. Halasz and Mayer D. Schwartz. "The Dexter Hypertext Reference Model" in Proceedings of the Hypertext Standardization Workshop by National Institute of Science and Technology (NIST), January 1990. reprinted in Communications of ACM (CACM), 37(2), 30-39, [Online: http://www.acm.org/pubs/citations/journals/cacm/1994-37-2/p30-halasz/], February 1994.

[Hall 1996] Wendy Hall, Hugh C. Davis, and Gerard Hutchings. Rethinking Hypermedia, The Microcosm Approach. Kluwer Academic, Dordrecht, The Netherlands, 1996.

[Handle 1999] "Handle System: A Persistent Global Name Service: Overview and Syntax" Internet Draft, 16 January 1999. http://www.handle.net/draft-sun-handle-system- 01.html, 1999.

[Lassila 1999] Ora Lassila and Ralph Swick (editors), "Resource Description Framework (RDF) Model and Syntax Specification" World Wide Web Consortium Recommendation, [Online: http://www.w3.org/TR/REC-rdf-syntax/], February 22 1999.

[Maurer 1996] Hermann A. Maurer. Hyper-G now Hyperwave : The Next Generation Web Solution, Addison

Wesley Longman, ISBN 0≠201≠40346≠3, 1996.

[McKelvie 1998] David McKelvie, Chris Brew , and Henry Thompson, "Using SGML as a Basis for Data-Intensive Natural Language Processing" Computers and the Humanities , vol. 31 , no. 5, pp. 367≠388, 1998 . http://www.ltg.ed.ac.uk/ódmck/Papers/chum.ps, 1998.

[Mealling 1999] Michael Mealling and Ronald E. Daniel. Resolution of Uniform Resource Identifiers using the Domain Name System, [Online: http://www.ietf.org/internet-drafts/draft-ietf-urn-dns-rds-01.txt], 1999.

[Nelson 1987a] Theodor Helm Nelson. Computer Lib/Dream Machines (second edition). Redmond, Washington: Tempus Books of Microsoft Press, 1987.

[Nelson 1988] Theodor Helm Nelson. "Managing immense storage" in Byte, 13(1), 225-238, January 1988.

[Pemberton 1999] Steven Pemberton (editor). XHTML 1.0: The Extensible HyperText Markup Language. A Reformulation of HTML 4.0 in XML 1.0. W3C Working Draft 4th March 1999, Cambridge, Massachusets: World Wide Web Consortium. [Online: http://www.w3.org/TR/WD-html-in-xml], 1999.