Artificial Intelligence Applications Institute

PLINTH, HTML and the World Wide Web

Andrew Casson

AIAI-PR-56

July 1994

Artificial Intelligence Applications Institute
The University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland

Copyright © 1994 by Artificial Intelligence Applications Institute, The University of Edinburgh

PLINTH, HTML and the World Wide Web

Andrew Casson
Artificial Intelligence Applications Institute
University of Edinburgh

July 1994

1. Introduction

PLINTH, the Platform for Intelligent Hypertext [1,2] is a domain independent *expertext* [3] shell which supports the processes of authoring and reading technical documents. Its data model is semantically enriched hypertext, where nodes and links in the hyperdocument can be assigned to user-defined taxonomic types and tagged with named attributes. The types are used to mark the function of each node of text (*e.g.* **introduction**, **background**, and **note** for almost all documents, **requirement** and **scope** in regulatory codes [1], etc.) and to make explicit the structural, logical and rhetorical links between them (*e.g.* **part-of**, **next**, **annotates** and **applies-in**). Named attributes can be used *e.g.* for classifying and indexing nodes, marking the strength of links, including information on the author of a node and its revision number, and so on. PLINTH also includes a rule-based network traversal tool, which provides the reader with automatic, intelligent navigation and consultation of semantically rich hypertexts, and supports the author with schemes for their principled creation and extension.

This paper and the earlier paper *The Use of Object-Oriented Databases in PLINTH* [4] look at solutions to problems with handling the potentially huge hypertext networks used by PLINTH. These problems fall into two categories.

Firstly, there are limitations on main memory, which restrict the size of data structures that can be handled and may make it difficult or impossible to work with very large documents in their entirety. We propose [ibid.] the use of object-oriented database (OODB) technology, which provides efficient and well-structured direct access to data in fast secondary storage (*e.g.* hard disk), as a potential solution to this problem.

OODBs do not, however, help with the second problem, which is that even the secondary storage available may not be able to cope with the vast number of documents to which a particular application may require access. For example, large commercial, governmental and academic organisations typically have whole libraries of technical documents necessary to their everyday functioning. Even if this could all be represented on-line, each document may well reference hundreds of other relevant works, which are not in the library but to which on-line access would be very useful. The largest organisations typically have many departments, each with its own library of technical reference material. If a document is needed by another department, then either it has to be physically transported between sites, or multiple copies must be stored—neither way is very efficient.

In fact, there are two obvious answers to the problem of lack of on-line storage: (i) obtaining more storage space, and (ii) storing less data. Procedures for (i) being well-known, we will concern ourselves with (ii). What we really mean by 'storing less data' avoiding local storage of documents that can be accessed electronically, on demand, from an external site. The external site may be another department of the same organisation, or a library on the other side of the world. Ideally, only the publisher of a book, manual or report would ever have to store a copy of it. Those wishing to use the document (and licensed to do so) would be able to transfer it over the network, page by page, as required.

This situation is likely to become more and more the actual case in future, and the basic technology is here already, in the form of the rapidly expanding Internet-based hypermedia network, the World Wide Web. This paper discusses the possibilities for integrating PLINTH with 'the Web'.

2. The World Wide Web

The World Wide Web (WWW) [5] began as a project at CERN in Switzerland to make sharing documentation between their research partners easy and efficient. In the last couple of years, however, it has grown to truly world-wide proportions, and continues to grow at a phenomenal rate, with dozens of new academic, commercial and governmental WWW sites appearing every day at the time of writing.

2.1 How does the WWW work?

The WWW is a essentially a distributed hypermedia system which uses the vehicle of the Internet, the vast international network of networks made up of hundreds of thousands of computers and accessible to millions of users. The basic idea is for each WWW site to store its publications, information about its research, academic or commercial interests and activities, and any other 'resources' that might be useful to others, in the form of hypertext documents marked up using **HTML** (see below). The site then runs a *server* program which accepts electronic requests for document pages[1] from *clients* elsewhere on the Internet, using **http**, the hypertext transfer protocol [6]. The server transfers the requested page to the client where it can be viewed using a special *browser* program (*e.g.* the widely used NCSA *Mosaic*) that sends http requests and displays the results. Each retrieved page may have embedded pointers to other pages, and activating one of these (*e.g.* by clicking on a highlighted word or phrase in the text with the computer's mouse) causes the client software to send off a new **http** request, possibly to an entirely different server.

2.2 HTML: the hypertext markup language

HTML is the standard ASCII markup language for WWW hypertext documents, and all WWW browsers must be able to handle it. HTML provides a simple format for

[1] a 'page' is the accepted term for a single **html** file, which may contain a document or document section of any size; it does not (necessarily) correspond to a displayed document page.

describing interlinked information resources, by means of *tags* embedded in texts, and it can be used to represent structured documents with inlined graphics, menus of options, hypertext views of existing bodies of online information, database query results, and many other types of information [7].

HTML is defined in terms of the ISO Standard Generalised Markup Language (SGML [8,9], which is a language for describing structured document markup schemes.[2] Every SGML document starts with a header consisting of a declaration and prologue, which describe both what constitutes an element in the ASCII data that follows, and how those elements may legally be combined (*i.e.* the 'morphology' and 'syntax' of the data and markup tags, to use a linguistic metaphor). An SGML *parser* program will check that the following text and markup tag sequence (the document *instance*) is well-formed with respect to the initial header. However, since all HTML documents share the same SGML header (by definition) this does not need to be included in every HTML resource on the WWW, or transmitted as part of every http transaction; instead it can be built into the client browsing software.

It is important to note that HTML markup is not intended to describe the exact visual appearance of a hypertext document (*e.g.* specific fonts, point sizes, paragraph layouts and margin widths). Rather it describes the *logical* structure of the document (*e.g.* emphasised text, headings, bulleted lists), and it is up to HTML browsing software like NCSA *Mosaic* to decide how best to present this to the user.

In any case, the most important feature of HTML is not its ability to describe the *internal* structure of a document, but its facility to embed cross-references to other WWW resources. This is done by means of *link anchor* tags, which specify the target resource by means of a *universal resource locator*, or URL. Full details of URL syntax are available in [10], but basically a URL has the form:

protocol://location/file#destination

where *protocol* is **http**, **ftp**, or **gopher** (see 2.3 below) *location* is the Internet machine address (*e.g.* **www.aiai.ed.ac.uk**), *file* is the pathname of the resource on the remote machine (*e.g.* **/~andrewc/home.html**) and the optional *destination* is the name of a *source anchor* tag marking a point within the target file.

Again, it is up to the WWW browsing software to decide what to do with this information. NCSA *Mosaic*, for example, highlights the word or phrase with which the link anchor is associated, and responds to a mouse click on the highlight by loading the resource and displaying it, whereupon the process may be repeated for links from the new page.

[2] Though the assertion that 'HTML is a subset of SGML' is frequently encountered, this is no more true than saying that a program is a subset of the language it is written in.

2.3 Other WWW data formats and access protocols

The resources stored at a WWW site may not all be in **html** format, and the **http** server can handle files of many different types, including plain (unformatted) text, **PostScript** ready for laser-printing, diagrams and pictures in **X bitmap**, **gif** and **jpeg** encoding, **mpeg** movies, and various audio formats. The server detects the file type automatically and reports it to the client, using the MIME (Multipurpose Internet Mail Extensions) [13,14] resource classification, before sending the data itself.

In addition, there is a huge amount of information on the Internet that is available not through **http** servers, but through other data transfer protocols and programs such as **ftp** and **gopher**, which have been around much longer than the WWW. Therefore **ftp** and **gopher** access have been integrated into the WWW data model, as we saw in the description of URLs above.

3. Accessing WWW resources from PLINTH

To connect PLINTH to the WWW then, we need to add three things: (i) a communication module for formulating and sending **http**, **ftp** and **gopher** requests and reading the response; (ii) routines for formatting and rendering the textual and graphical contents of the **html** pages and other multimedia resources that are returned; and (iii) a means of accepting user input to determine which resource to load next (*e.g.* via mouse-clicks as in *Mosaic*).

3.1 Communications

Although it would be possible to write a complete communications module for PLINTH from scratch, it would be a very complicated and time-consuming task requiring detailed knowledge of the protocols used and experience in writing networking software to do a proper job. Also, the software would have to keep up with any future revisions of the protocols. Fortunately, there is no need to expend such effort, as there already exists in the public domain an excellent, well-tested WWW communications package, **libWWW** [11] produced by the CERN team and used in NCSA *Mosaic*. The package is written in C, so it should be quite straightforward to write a Pop11 interface to it for use in PLINTH, and upgrading this interface to work with new releases of **libWWW** should be a minor task. The only potential drawback with using any public domain software is the possible future commercial exploitation of PLINTH, although at the moment the use of **libWWW** does seem to be completely unrestricted.

3.2 Displaying WWW resources and interacting with the user

The display of non-HTML resources presents no problems for PLINTH. Plain text, X bitmaps and GIF images can be viewed in TOME, and like other WWW browsers such as *Mosaic*, PLINTH can rely on standalone, mostly public domain programs

(which the client site is responsible for installing) for displaying other resource types, *e.g.* **xv** for images, **GhostView** for PostScript, **AudioFile** for sound, and **mpeg_play** for movies. The user may specify a mapping from MIME types to programs to override the default viewers if needed. The auxiliary nature of these resources makes this approach quite acceptable.

HTML itself could be displayed in TOME using the public domain **HTMLWidget** [12] an X Toolkit widget used by NCSA *Mosaic*. This widget takes an HTML string as the value of its **XtNtext** resource and displays it, formatted, in a window (although routines for rendering inlined images are not built in). However, unlike **libWWW**, using the **HTMLWidget** could really only be classed as a 'quick and dirty' method: it is useful for getting WWW access up and running as soon as possible in order to demonstrate its usefulness, but it has serious drawbacks. The user would be faced with two kinds of TOME window with rather different looks and behaviour: one editable, the other not; one where buttons (*i.e.* source anchors) are true nodes, one where they are not; one where internal reference fragments (*i.e.* destination anchors) are highlighted regions, one where they are a single point in the text; and two distinct text selection and highlighting schemes. In short, it would be a confusing mess. For something that could become a very important central feature of PLINTH, a less superficial approach, more deeply integrated with the main system, is highly desirable.

Two solutions are available. Firstly, and more easily, we can add conversion routines to PLINTH for pre-formatting the HTML into a TOME buffer (keeping the distinct appearance of bold and italic text but not the variety of font sizes) which can then be viewed *and edited* in a regular TOME window. HTML link anchors can be converted into proper button nodes which have a URL slot rather than a normal reference link to another node. The TOME representation can be converted back to HTML if the file is saved locally. Inlined images could be displayed simply as cross-reference buttons to separate windows (*i.e.* not inlined at all), or image display facilities could be added to TOME (a major task, since the Poplog XpwScrollText widget that TOME uses for its basic display and input of text does not support graphics, and would have to be re-implemented from scratch).

The second, and much better, solution would be to use a version of HTML as TOME's *native* text format, and avoid the conversion altogether. The next section of the paper looks at the possibilities for achieving this.

4. Using HTML as TOME's native text format

In its current incarnation, HTML does not provide a sufficiently rich text markup scheme for it to be used to represent PLINTH documents. Even its text formatting facilities are very basic, and not really up to describing pages with complex tables and annotated diagrams such as are found in the building standards document [15] to which PLINTH is initially being applied. However, a new version of HTML called

HTML+ is currently being developed[3] which will provide tables, subscripts and superscripts, and floating figures.

The greatest problem with HTML as it stands is that the ability to tag nodes and links with semantic types and attributes is completely lacking. A very limited form of link typing with just a few fixed types has been proposed for HTML+, but this is of little use to PLINTH. 'Semantic headers' have also been suggested for even later versions, but details of what these might contain are as yet unavailable.

However, defining a new *superset* of HTML+ for PLINTH's own use, with all the necessary semantic tagging features, could be a worthwhile development. SGML is certainly powerful enough to support this. For example, the text of a node could be marked up as in the following example:

<node type=section **title="**Regulation 26**" display=**window**>**

<node type=requirement **title="**Regulation 26.1**" display=**region **id=**26.1**>**
text of Regulation 26.1
</node>

<node type=scope **title="**Regulation 26.2" **display=**region **id=**26.2**>**
text of Regulation 26.2
</node>

<link type=application **from-id=**26.1 **to-id=**26.2**>**
</node>

which represents a *section* "Regulation 26" containing a *requirement* clause and a *scope* clause (to be displayed as highlighted regions within the section window) which have an *application* link between them. The advantages of using this expertext markup language (ETML) are:

- Since ETML is a superset of HTML(+) then PLINTH can read and format the latter automatically[4].
- If HTML(+) is extended to allow *user-defined* tags that can be ignored by browsers that do not recognise them, as has also been proposed, then *Mosaic* and suchlike will be able to format PLINTH data without being upset by the semantic tags.
- SGML markup can be used for document analysis as well as display. Generic SGML tools for doing this will be able to work with PLINTH data.

---

[3] certain HTML+ features are already supported by NCSA *Mosaic* and used on the WWW, such as fill-in forms.

[4] As HTML(+) evolves, ETML will need to be revised accordingly in order to maintain compatibility. However, maintaining compatibility is an issue whatever the type or level of support for HTML in PLINTH.

Of course, SGML is a textual data format intended for file-based storage and transmission. While it can be used 'raw' as TOME's internal representation of the marked-up text (replacing the current system in which each 32-bit character cell can be marked individually with its visual properties and start/end markers for nested nodes) and edited invisibly through a WYSIWYG interface, the ETML data would still need to be 'compiled' into an efficient internal framed-based or object-oriented representation (*e.g.* using the CONNEKT library as now) before being manipulated by other parts of the PLINTH system. For example, navigation rules would still need to view a node as an object with a type and various named attributes, rather than a just string of text from which the information has to be extracted every time it is needed.

5. Summary and conclusions

It is probably no exaggeration to say that the technology and ideas currently being introduced by the World Wide Web are just the start of a fundamental transformation in the way information and documentation is produced and distributed. Rather than printing thousands or even millions of copies of a publication, the technology is emerging which will support the ability to store a single on-line copy, which can be retrieved and viewed on screen, on demand across international computer networks.

A general purpose technical documentation system like PLINTH cannot afford to ignore these developments, and therefore a PLINTH interface to the WWW would be a very important addition to its functionality. In this paper we have examined the formats and protocols used in WWW information interchange, and discussed the various levels of support that PLINTH can provide for them. A great deal of work still needs to be invested in researching, developing and implementing these ideas, but the potential return could be very great.

References

1. Casson, A. (1993). "PLINTH: Integrating Hypertext, Semantic Nets and Rule-Based Systems in an Expertext Shell for Authors and Readers of Regulatory Documents", in Mayfield, J. and Nicholas, C. (eds), *Proceedings of Workhop on Intelligent Hypertext CIKM '93*, Arlington VA, November 1993.

2. Casson, A. and D. Stone (1992). "An Expertext System for Building Standards", in *Proceedings of Workshop on Computers and Buildings Standards*, Montréal, Canada, May 1992.

3. Diaper, D. and M. Beer (1990). "Headed record expertext and document applications", in *Proceedings of Hypertext Update*, Unicom Seminars Ltd, London.

4. Casson, A. (1994). "The Use of Object-Oriented Databases in PLINTH". AIAI-TR-145, Artificial Intelligence Applications Institute, University of Edinburgh, March 1994.

5. Berners-Lee, T. (1994). "World Wide Web Initiative". CERN, Geneva. *http://info.cern.ch/hypertext/WWW/TheProject.html*

6. Berners-Lee, T. (1994). "HTTP: A protocol for networked information". CERN, Geneva. *http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html*

7. Berners-Lee, T. and D. Connolly (1994). "Hypertext Markup Language (HTML)", CERN, Geneva. *http://info.cern.ch/hypertext/WWW/Markup/HTML.html*

8. ISO 8879 (1986). "Information Processing Text and Office Systems Standard Generalized Markup Language (SGML)".

9. Bryan, M. (1988). "SGML, An Author's Guide to the Standard Generalized Markup Language", Addison Wesley.

10. Berners-Lee, T. (Ed.) (1994). "Uniform Resource Locators". CERN, Geneva. *http://info.cern.ch/hypertext/WWW/Addressing/URL/Overview.html*

11. Berners-Lee, T., J.-F. Groff and H. Frystyk (1994). "Common WWW Code Library". CERN, Geneva. *http://info.cern.ch/hypertext/WWW/Library/Status.html*

12. [author unknown] (1994). "HTML Widget". National Center for Supercomputing Applications. *http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/htmlwidget.html*

13. Grand, M. (1993). "MIME Overview".

14. Borenstein, N. and N. Freed (1992). "MIME (Multipurpose Internet Mail Extensions): Mechanisms for specifying and describing the format of Internet message bodies". Network Working Group Request for Comments 1341, June 1992.

15. The Scottish Office (1990). "Technical Standards for compliance with the Building Standards (Scotland) Regulations 1990", Edinburgh, Scotland.