Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

The MADE help system

M. Haindl and M.M. de Ruiter

Computer Science/Department of Interactive Systems

CS-R9528 1995

Report CS-R9528
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

The MADE Help System

Michal Haindl and Bèhr de Ruiter
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

email {*mh,behr*} *@cwi.nl*

Abstract

MADE is the acronym for the ESPRIT project 6307, whose aim is to develop an object oriented multimedia application development environment. As part of this project the MADE help system is designed to be a distributed hypermedia system with additional support for run-time object monitoring and contextual help.

AMS Subject Classification (1991): 68N99
CR Subject Classification (1991): D.1.3, D.1.5, H.3.5, H.5.1, H.5.2, I.3.2
Keywords & Phrases: MADE, multimedia, help architecture, WWW browser, object monitoring.
Note: This paper is accepted for presentation on the EUROGRAPHICS'95 Conference in Maastricht, 1995.

1. Introduction

Multimedia applications form a booming part of present-day computer industry, partly because they represent a natural synthesis of available digital data processing techniques and so can benefit from its results (signal processing, image processing, computer graphics, pattern recognition etc.) and partly because computer manufactures hope to use them to expand their sales.

Most of the available commercial multimedia systems are closed; they do not easily allow their users to modify their functionality or to freely combine their separate program elements in an un-envisaged way. Unfortunately, it is almost impossible to define a closed programming environment which encompasses all useful multimedia techniques and applications. The obvious answer to this challenge is to use object-oriented techniques: services are offered in the form of objects and a programmer can create any sophisticated tool by combining these objects with application-dependent supplements. To realize this, the MADE (Multimedia Application Development Environment) project aims to design and implement such a portable object-oriented development environment for multimedia applications. The developed software, based on the C++ programming language, runs on several UNIX platforms as well as in a Windows-NT environment. The project software can be divided into two distinct functional levels, the toolkit level containing fundamental multimedia objects, and the utilities level, which provides more complex functionality by combining elementary toolkit objects. For a general overview of the project[11], the underlying object model (the following section and Arbab et al[1]) or the authoring environment[10] see the relevant references.

Modern help systems should be built as hypertext systems, ideally with full multimedia support as well. Such expressive power can significantly increase the information value of the system or even bring information difficult or impossible to describe in a traditional plain text help systems to the user. Data distribution is another important aspect for generally applicable software systems like the MADE system. Nobody can create a local information database with an amount of information comparable to the amount of information already available on the Internet without even taking into account its present rapid growth. Easy maintenance, platform independence, software support and wide acceptance are other important requirements.

The MADE help system, which fulfills the above requirements for modern help systems, is a restricted distributed hypermedia system using HTML [3] (Hypertext Markup Language) (and some HTML+ [13] markups), which is based on SGML [4] (Standard Generalized Markup Language) that was developed for the World Wide Web project. Unfortunately this markup language lacks the means to describe time relations. It is only possible to passively invoke dynamic data viewers without any possibility of mutual synchronization. Unlike most document formats, markup languages describe the logical elements of a document, but leave the rendering specification (like font name, point size, margins, etc.) to the rendering browser. HTML+ departs slightly from the pure markup language idea by allowing certain kinds of rendering information.

Although HTML possesses some disadvantages, like the lack of synchronization functionalities and that rough format specification is not yet formally standardized, its main advantages are the possibility to access dynamic world-wide distributed data, easy sharability of help data, platform independence, cross-platform development, and a large existing software support. Another advantage is its de-facto world-wide standardization, which will sooner or later result into a formal ISO standard as well.

All MADE help documents and media data accessed via these documents can be distributed throughout the Internet and can be reached via FTP, HTTP or some other remote servers.

The MADE help system is structured as four toolkit objects covering the basic help functionality: manual page presentation, contextual help, data presentation and personal annotation; and two utilities [5],[6], [7]. One of the utilities provides the MADE WWW hypermedia browser. The other object monitoring utility, enables the observation of run-time object behaviour and to record this information in an audit-trial file. The fundamental difference between the object monitor and a debugger is that the former does not interfere with the monitored object and its data (i.e. does not alter the object's behaviour) which is a crucial requirement in a multi-thread application environment like MADE whereas a debugger is necessary intrusive. The object monitor represents rather a software realization of an oscilloscope attached to some potentially problematic data place.

2. Object Model

The MADE object model was defined[1] to ensure the smooth cooperation between objects in the MADE environment and to provide a unified conceptual approach to some technical issues raised by multimedia programming in general. The object model has two distinctive features: active objects and delegation.

Objects may be active, i.e. they can have their own thread of control (within the shared address space of the same UNIX or WINDOWS NT process). However because of efficiency reasons they are realized as four different types of objects:

1. *active object* - own thread of control and responsibility for serving messages, delegation functionality;
2. *mutex object* - mutually exclusive execution in the thread of its caller, delegation functionality;
3. *unprotected object* - unprotected (no exclusivity protection) execution in the thread; of its caller, delegation functionality
4. *C++ object* - unprotected execution in the thread of its caller, no delegation functionality.

The delegation concept applies to an object's methods. An object may dynamically delegate some or all of its behaviour (i.e., the messages it serves) to any number of other objects, which then act on its behalf.

The object model is created in the form of an extension of C++, called mC++, an mC++ translator, which generates a set of C++ classes, and a run-time library; the intermediate level can also be accessed directly by programmers if they do not wish to use another programming language.

3. Help Toolkit

The help toolkit objects provide the basic help functionality to present explanatory information to the user and enable the user or any MADE object to create annotations. As mentioned in the introduction, these objects provide for manual page presentation, contextual help, data presentation, and personal annotations. All four objects are mutex objects, so they can be safely shared by different callers and they use the services of the browser utility for presentation of their data. The browser's own functionality (hyperlink following, format transformations, annotation, printing etc.) enhances user comfort and adds valuable capabilities to the use of the toolkit.

The following sections describe each of the help toolkit objects.

3.1 Manual Pages

MADE online reference manuals and technical reports are maintained in a hypermedia format. The manual page object has three main capabilities: document presentation, keyword search and presentation of a hypermedia document sequences. It has a straightforward interface that allows for the presentation of HTML documents, beginning from a specified anchor or a UNIX troff formatted manual page.

Note that HTML does not support complicated mathematics or graphical structures, but these can always be presented as embedded images. This solution is only limited in its usage for such elements as hot-spots.

The keyword search function locates all occurrences of the required keyword in a document and finds the nearest possible anchors in front of their text locations. These anchors then serve as starting points for document presentation based on keyword search results.

Finally, it is possible to specify a sequence of unrelated documents (without mutual hyperlinks) or their pieces for orchestrated sequential stream presentation. These documents can be hypermedia documents, plain text documents or dynamic media data.

3.2 Context Sensitive Help

The purpose of context sensitive help is to offer the user advise on how to recover from erroneous input that is dependent on the context of some interaction. This concept differs from usual meaning of the term context sensitive help i.e. help information related to particular context of an application. Obviously our mechanism is more general and includes also the usual meaning of this term (input is a help request, trigger test is always positive, and help information has one level only).

The implemented system is hierarchical, where the first level handles recovery from mistyping errors with a short explanation, the second level offers a tutorial explanation by jumping to an appropriate hypermedia document location. Finally, the third level is the warning that due to systematically incorrect input an irregular behaviour of the system should be expected. The selection between levels is automatic - the first error occurrence is assumed to be mistyping , the second occurrence requires explanatory level, etc. This automatic behaviour can be modified for example, to specify a four level help with two explanatory hypermedia documents. A message on the first level can be any combination of two strings and one real number.

3.3 Data Viewers

Data viewers, as the name suggests, add to the toolkit presentation a specified part of some basic data (integer, float, character) vector or matrix in a user friendly form. The viewers support three output options: they can direct content to a window, a disk file or both.

Data viewers can automatically determine the format of output based on the range of numerical values to be presented and available display extent. By default, this extent corresponds to an A4 sized page.

The example of possible output:

```
float submatrix —[10,20],....,[10,90]— printed in 11 parts
                ..............
                —[12,20],....,[12,90]—
1-th part columns [20-26] ———————
3.01e+01 3.11e+01 3.21e+01 3.31e+01 3.41e+01 3.51e+01 3.31e+01
3.11e+01 3.21e+01 3.31e+01 3.41e+01 3.51e+01 3.61e+01 3.31e+01
3.21e+01 3.31e+01 3.41e+01 3.51e+01 3.61e+01 3.71e+01 3.21e+01
```

3.4 Personal Annotations

The fourth help toolkit object permits annotations to be defined and associated with any other MADE object. These are stored by the help tool together with references to the annotated object. Later they can easily be used, for example, for multimedia document editing. MADE annotations are persistent, contain a hidden creator object signature, can be defined automatically or interactively, and are stored in separate files. The interactive annotation dialogue directly operates in the annotation window of the browser (see annotation window on Fig.4.2) without the necessity to proceed from the main window.

An annotation can be also appended to an existing annotation file provided that both share the same signature. This signature is assumed to be a string, allowing any keyword or password shared by several MADE objects to be used. In such cases objects clustered can append their messages to their corresponding seed annotation files.

Object annotations are formally related to the HelpAnnotation.html document, whose sole purpose is to permit visualization of the annotations. They can be presented or edited independently from their creator like standard annotations belonging to any other HTML document.

4. Help Browser

The help browser (Generic Help Facilities) as specified in[5], is implemented[9] as a controllable WWW browser, which is used to present selected pieces of help information from a potentially world-wide distributed MADE hypermedia-style document. The browser is a mutex object and represents the core part of the Made Help System[6]. It can also be used as a stand-alone program similarly to other WWW browsers (e.g. Mosaic, Netscape, Lynx etc.).

The browser has features common to other WWW browsers, like hyperlink following, remote date retrieval, external data playing, personal annotations, history, keyword search capability, hot-list etc. A plain text file can be displayed or a HTML file can be converted into a plain text or PostScript file. The format manager uses the communication modules to load the document from the network as appropriate, depending on its URL (Uniform Resource Locator). It can also decide on the format of a file from its name or completes the URL if the address is relative or pointing to a directory instead of a file. The history module records and replays on request the documents which the user has visited. This history is identical for all open browser main windows.

Besides these common features it has also some unique ones. The browser is fully remotely controllable [9]. It is possible through the browser control protocol to dynamically modify its behaviour. For example, it is possible to temporarily suspend some browser activities, change a meaning of some user action, use the running browser to retrieve and display a new document or use some browser

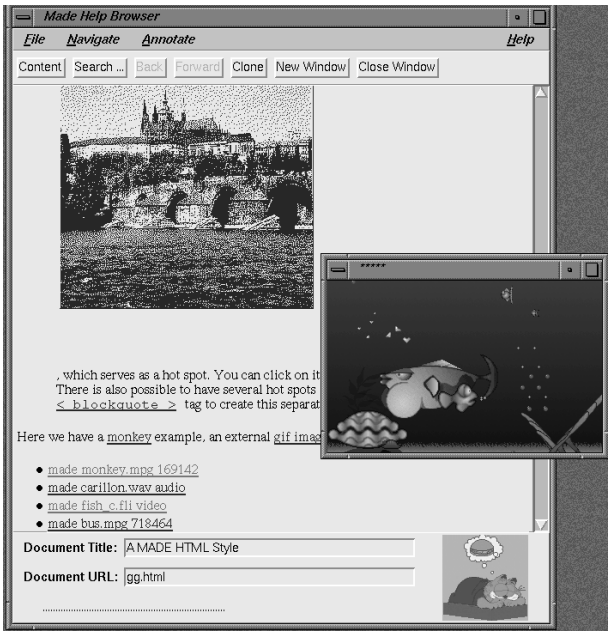

Figure 4.1: Link following with a video mpeg file.

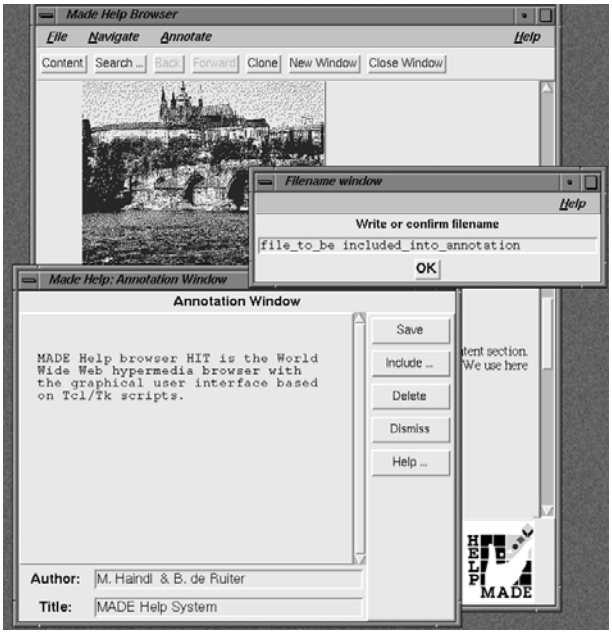

Figure 4.2: Document annotation.

functionality (e.g. format conversion, data printing, etc.) for another application purpose. One simple application of this feature can be access restriction for some users. Others are ignoring external viewer spawning if the network is overloaded, and data content dependent actions. The remote control protocol enables also a dynamic reconfiguration of the user interface. It is possible to iconify the browser, to make it invisible, to remove or restore some browser functionalities e.g. elements in pop-up menus etc.

The browser can directly use MADE specific media data (made server[5]). However this relies on the use of MADE media viewers and consequently DBO object (a generic interface between the MADE environment and external databases[2]) in conjunction with some connected database. The browser does not use their synchronization mechanism; it only retrieves and plays single media objects. Using MADE media viewers enables us to hide any specific data formats but their use is currently limited to local network data access only.

The browser can simultaneously present several independent hypermedia documents, each one in its own main window.

Distributed data is supported for http, gopher and ftp servers and different media formats can be handled using public domain viewers (txt, text, tex, ps, gif, jpeg, au, snd, aiff, aifc, dvi, tiff, mpeg, mime, xwd, movie, evlm, rgb, rtf, gz, Z etc.) see Fig.4.

The graphical user interface is written in Tcl/Tk and so it is platform independent. The browser uses twelve different user interface windows, i.e., main browser window, annotation window (Fig.4.2), keyword search window, navigate pop-up menu window, file menu window, history window, annotation menu window, hot-list window, URL window, file saving and converting window, print window and the file input window. The standard browser usage starts from the main window, but the browser can also operate on a functional subset of these when used with some Help toolkit objects, for example. The canvas part of the main window itself consists of several opaque windows used to display formatted text and embedded still images.

The browser by default uses its own MADE formats for the global history, hot-list and personal annotations. It is also possible to configure the browser to produce or read the corresponding Mosaic data formats.

5. Object Monitoring

The OBM utility provides a mechanism for observing an object's behaviour at run time without interfering with its operation. This information can be recorded in audit-trial files. The fundamental philosophy and specification of the OBM is given in [5]. Limitations introduced by C++ prevent the object monitor from monitoring single messages. Instead data changes resulting from these messages are observed. The OBM uses a delegation mechanism for dynamic connection and disconnection between one or several data sources (monitored objects). A control object creates the OBM object and attaches it with some object to be monitored, later other monitored objects of the same class can be connected or disconnected to this OBM object.

The OBM itself[6] has its own graphical user interface based on Tcl/Tk[12] which contains six different user interface windows. The global selection window (Fig.5.3) can be used to apply globally monitoring options to all monitored data, i.e. presentation output (display, file or both), monitor all data, selected data or data changes only, and finally to select monitoring activity. The monitoring activity can be either permanent, reacting to every request, or frequential, reacting only to requests that fit into a specified range of activity. The data selection window enables fine grain monitoring data selection as seen in Fig.5.4.

The OBM and DBO (database object) share the same concept of object data types (elementary data types and data structures created by a snapshot function). Monitored data is assumed to have

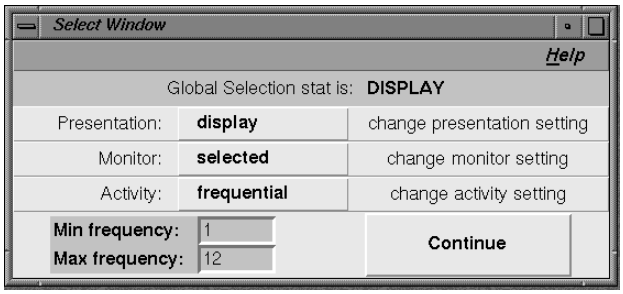

Figure 5.3: The global data selection window.

a tree structure, where single nodes can represent elementary data types, pointers, strings, lists, sets, or objects. The current tree node button on this window actives the pop-up window showing the path from the current node to the root node of the data tree. This path serves as a data structure orientation tool as well as a browsing tool. All displayed nodes have labels that serve as hot-spots to the corresponding data tree nodes.

When data selection has been completed and the chosen presentation mode includes display, then monitored data is presented in the data monitoring window, Fig.5.5. This window presents all selected data on the same tree level, subsequent child nodes can be viewed by selecting the corresponding node's display buttons. Every monitoring window is capable of storing its local data, even if it was not previously designated for file storage. Data selection can also be adjusted in this window.

The implemented OBM object has the following main features:

- Every MADE object which has implemented a snapshot function can be monitored. The snapshot function represents the maximal extent of monitorable information and is specified by the object's author. The snapshot function is also used for database object data storage[2].
- One monitored object can have several object monitors, each one with its own specified data selection.
- One object monitor can be attached to several data sources (monitored objects), so it is possible to compare data from several sources. The monitored objects have to be derived from the same class and share the same monitoring data selection.
- New data sources can be dynamically added or old ones dynamically removed from an attached object monitor.
- Monitored information can be recorded in a file and/or presented in a window.
- File storage and display mode have mutually independent monitored data selections.
- Data selection choice can be stored in the database for later use.
- Data monitoring can be triggered from an application or from the monitored object itself.

The OBM can monitor all other help objects including another OBM objects similarly with most of other MADE objects.

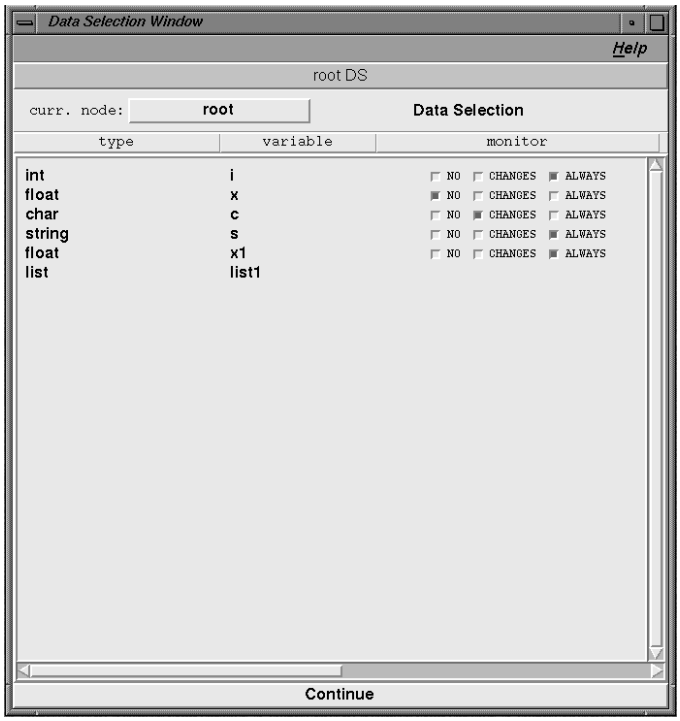

Figure 5.4: The data selection window.

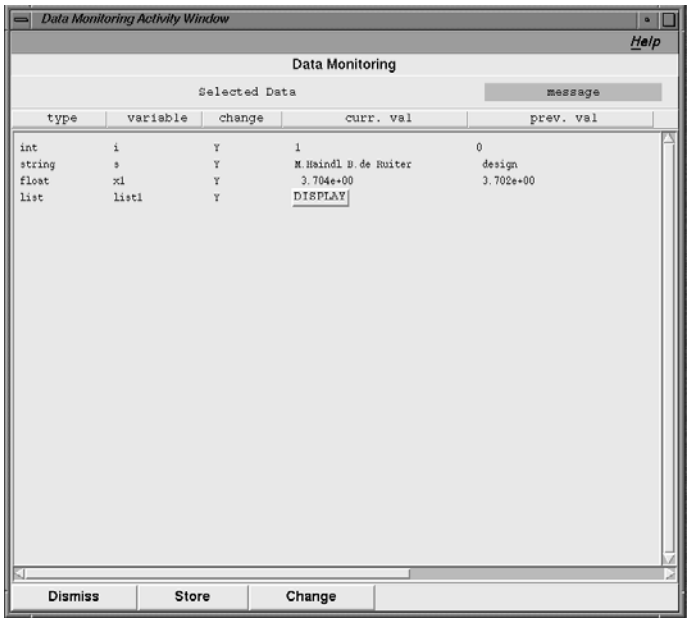

Figure 5.5: The data presentation window.

6. Conclusions

The help system described in this paper represents a powerful information support environment for MADE users. The system fulfills all of the requirements for a modern help support environment using powerful WWW architecture. This approach not only enables the use of local help files, but also provides access to the unlimited and constantly growing resources of a global information space. Help information does not have to be prepared in advance by a multimedia document author. Users can simply use every Internet querying system like Archie, WAIS or Gopher to obtain the desired information. Such information can have a form that is any combination of text, image, video sequence or audio. Single pieces of help information can be world-wide distributed over the Internet; their usage is only limited by corresponding retrieval delays.

This paper has also described the help system object monitoring capability. The internal run-time state of objects can be monitored for debugging purposes or it can be used for data access monitoring. Such information usage statistics can be useful in improving hypermedia document structure, in process monitoring, in automatic examination systems, and many other applications.

The contextual help function offers effective structured help information relevant to an interactive situation, and also enables a robust learning or examining application to be constructed as well.

Our system is one of the first fully distributed help systems which is platform independent, running on UNIX systems and on Windows-NT platforms. It clearly outperforms the $WinHelp^{TM}$ functionality as the only existing optional online help standard. There is no necessity to compile help files. We consider this to be a serious drawback of the $WinHelp^{TM}$.

Online help information may also be reused to produce printed output. However, to print a general hypermedia document is a difficult task. Even ignoring the problem of printing dynamic media (audio, video), the static data itself may need to be restructured when there is no obvious mapping between a general hypertext graph structure and a standard book (report) structure. Print hypertext using some predefined conventions will hardly result in satisfactory output.

The MADE help system can be used as a hypermedia presentation tool for applications which do not require sophisticated synchronization capabilities like, for example, online tutorials, a tourist information kiosk, or a digital encyclopedia.

Acknowledgement

This work has been carried out as part of an EC funded ESPRIT research and development project called MADE (no.6307). The authors would like to thank their project colleagues for their comments during their work on the help system.

References

1. F. Arbab, P.J.W. ten Hagen, M. Haindl, F.C. Heeman, I. Herman, G.J. Reynolds, A. Siebes, "Specification of the MADE Object Model," Technical Report T/OM/S1, *Esprit Project 6307 MADE*, 1993.
2. C. van den Berg, F. van Dijk, "Specification of the Database Object," Technical Report T/DBO/S.4, *Esprit Project 6307 MADE*, 1994.
3. T. Berners-Lee, "Hypertext Markup Language," Internet Draft, 1994.
4. C.F. Goldfarb, "The SGML Handbook," Clarendon Press, Oxford, 1990.
5. M. Haindl, "Specification of the Help Object," Technical Report T/HEO/S.0, *Esprit Project 6307 MADE*, 1993.
6. M. Haindl, "Specification of the Object Monitoring," Technical Report U/OBM/S.0, *Esprit Project 6307 MADE*, 1993.
7. M. Haindl, "Specification of the Generic Help Facilities," Technical Report U/GHE/S.0, *Esprit*

Project 6307 MADE, 1993.

8. M. Haindl, B. de Ruiter, "Prototype Implementation of the Generic Help Facilities," Technical Report U/GHE/P.0, *Esprit Project 6307 MADE*, 1994.
9. M. Haindl, "Implementation of the Help Objects," Technical Report T/HEO/P.1, *Esprit Project 6307 MADE*, 1994.
10. M. Haindl, I. Herman, G.J. Reynolds, "Presentation Scheme," Technical Report U/PRS/S.0, *Esprit Project 6307 MADE*, 1993.
11. I. Herman, G.J. Reynolds, J. Davy, "MADE: A Multimedia application development environment," *Proc. on ICMCS* , Boston, pp. 184-193, 1994.
12. J. Ousterhout, "An Introduction to Tcl and Tk," University of California, Berkeley, 1992.
13. D. Ragett, "HTML+ (Hypertext markup format)," Internet Draft, 1993.