An Evaluation of the World Wide Web with respect to Engelbart's Requirements

Daniel W. Connolly

W3CMIT LCS

$Date: 1996/11/22 04:40:55 $

Status of This Document

This document is informational only. It does not specify any new specifications or protocols. The editor is working towards makin this document a W3C Note.

Abstract

The architecture and implementation of the World Wide Web are evaluated with respect to Douglas Engelbart's requirements for an open hyperdocument system. Engelbart's requirements are derived from experience in using computer-supported collaborative work (CSCW) to support large scale electronic commerce.

Keywords Hypertext Systems Web Design

An Evaluation of the World Wide Web with respect to Engelbart's Requirements

Daniel W. Connolly

W3CMIT LCS

$Date: 1996/11/22 04:40:55 $

Contents

1. Introduction

The World Wide Web is targeted at collaborative work: research, development, marketing, advertising, negotiation, sales, and support, if not more. in this way. It borrows many design principles from the research in computer supported collaborative work (CSCW), which showed how effective hypermedia systems can be for knowledge capture, knowledge exchange, and collaboration in general.

In comparison to even the early hypertext research systems, the Web is fairly simple and somewhat limited. And yet it is extremely widely deployed and exploited. Is the Web technology ultimately limited? Is it a passing phenomenon, or is it a viable long-term platform for collaborative work and electronic commerce?

In [Eng90], Douglas Engelbart condensed the results of many years of research in computer supported collaborative work into twelve requirements[ENG90]. This report evaluates the Web with respect to each of the requirements: has it been met? If not, can it? What are the obstacles, and what are the most promising developments in each area?

2. Essential Elements of an Open Hyperdocument System

Engelbart's research was directed at large scale knowledge work; for example, in the aircraft industry, the interactions between a major manufacturer and its contractors, subcontractors, and so on. Research and experimentation led to the following requirements:

1. Mixed Object Documents
2. Explicitly Structured Documents
3. View Control of Object's Form, Sequence, and Content
4. The Basic "Hyperdocument"
5. Hyperdocument "Back-Link" Capability
6. The Hyperdocument "Library System"
7. Hyperdocument Mail
8. Personal Signature Encryption
9. Access Control
10. Link Addresses That Are Readable and Interpretable by Humans
11. Every Object Addressable
12. Hard-Copy Print Options to Show Address of Objects and Address Specification of Links

3. Overview of Web Architecture

In order to evaluate the Web with respect to Engelbart's requirements, some background on the architecture is essential.

The Web is a hypermedia distributed information system. A distributed information system allows users to access (read, search, navigate) documents from a wide variety of computers connected via a network. A hypertext system is one where the documents have links in and between each other, so that readers are not constrained to a sequential reading order. Finally, a hypermedia system is one that integrates images, sounds, video, and other media with text in documents.

The Web incorporates a wide variety of information sources on the Internet into a coherent information system. The user experience on the Web is:

1. An initial document is presented to the user, consisting of text, graphics, sounds, animation, etc.
2. The user can choose one of the links in the document and request to visit the document on the other end of the link. The link specifies the address of another document.
3. The system retrieves the document at that address and presents it to the user, and the process repeats.

3.1. Protocols and Addressing Schemes

A URI, or Uniform Resource Identifier, is a name or address for an object in the Web[URI]. For example: http://www.w3.org/ ftp://ds.internic.net/rfc/rfc822.txt

Each URI begins with `scheme :`, where *scheme* refers to an addressing or naming scheme. For example, URIs beginning with `http:` refer to objects accessible via the HTTP protocol; `ftp:` refers to FTP objects, and so on. New schemes can be added over time.

There are a number of information retrieval protocols on the Internet: FTP, gopher, WAIS, etc. In the Web user experience, they behave similarly: a server makes a collection of documents available. A client contacts a server and makes a request to access one or more of the documents by giving a command telling what sort of access (read, write, etc.) and some parameters that indicate which is the relevant document (s). Finally, the server fulfills the request by, for example, transmitting the relevant document to the client.

HTTP is a protocol designed specifically for the Web[HTTP]. It is an extensible protocol with support for the widely used features of existing protocols, such as file transfer and index searching, plus support for some novel features such as redirection and format negotiation.

3.2. Data Formats

There is no one data format for all Web documents—each document may have its own format. In fact, it may be available in many formats.

Each data format has a name, or an Internet Media Type. Internet Media Types, aka content types, are part of MIME[MIME], a collection of enhancements to Internet mail to accomodate rich media. Most Internet Media Types are just standardized names for existing data formats; for example, `text/plain` for normal ASCII text; `image/gif` for images in GIF format, etc.

The MIME standard also introduces some new data formats. The `multipart/mixed` is a *compound* data format—it allows several pieces of media to be combined into one, for transmission via email, storage in a file, etc.

Hypertext Markup Language (HTML) is a data format designed specifically for the Web. It combines the features of a typical structured markup language (paragraphs, titles, lists) with hypertext linking features.

HTML is an application of SGML, the Standard Generalized Markup Language[SGML]. SGML is a technology for specifying structured document types. HTML is one such document type, but there are many others—as many as anyone cares to dream up. TEI, DocBook, and Pinnacles are just a few of the types of SGML documents used in the web[NCSASGML].

4. Evaluating the World Wide Web

This evaluation measures the World Wide Web against each of Engelbart's requirements, discussing strengths, weaknesses, and promising developments.

4.1. Mixed Object Documents

> to provide for an arbitrary mix of text, diagrams, equations, tables, raster-scan images (single frames, or even live video), spread sheets, recorded sound, etc.—all bundled within a common "envelope" to be stored, transmitted, read (played) and printed as a coherent entity called a "document."

The first Web documents contained only text, but support for icons and images was added to NCSA Mosaic in 1993. Since then, Web pages with mixed text and graphics have been the rule, and sound and video are not uncommon. As a recent development, tables are widely deployed. Support for equations is still essentially in development, and diagrams are generally limited to rasterized snapshots.

Exchange of spreadsheets and other rich data sets associated with proprietary software packages is supported to some extent, but its use is generally limited to small communities of interest who agree to use the packages.

This demonstrates that at least to some extent, the requirement for mixed object documents is met. But it is not met completely: the tools for composing mixed object documents are primitive, and many features of a comprehensive compound document architecture are lacking.

4.1.1. Authoring Tools

The intent of the original design of the web was that documents would be composed in direct-manipulation fashion from a rich set of media objects. The initial prototype was done on the NeXTStep platform, which allows drag-and-drop editing of rich text documents including raster images, encapsulated postscript images, sounds, diagrams, etc. The NeXTStep platform includes an architecture and a set of development tools for adding new object types to the mix available on the desktop.

System such as [Hyper-G] provide many of these features, and recently products such as [NaviPress] and [FrontPage] beginning to provide these features to the web user base at large.

4.1.2. Compound Document Architecture

The two technologies used to create compound documents on the web are URIs and MIME. Objects can be linked together by using their addresses. For example, the HTML A, IMG, FORM, and LINK elements specify URIs of linked objects. Links are typed to indicate various relationships between objects such as parent/child, precedes/succeeds, supports/refutes, etc. The MIME multipart facility allows several pieces of content to be combined into one entity. Support for typed links and multipart data is deployed only to a limited extent.

The combination of URIs and MIME supports the entire existing web information space. Still, compared with compound document architectures such as OpenDoc, OLE, Fresco, LINCKS[LINCKS], or HyTime, many facilities are lacking.

Facilities for diagrams, equations, screen real-estate arbitration, event management, versioning, transactions, link indirection, and aggregation have been developed for various web applications, but there are no standard facilities. Standards for such facilities might result in a critical mass of shared of technology that would make feasible classes of applications that were previously too costly.

4.2. Explicitly Structured Documents

> where the objects comprising a document are arranged in an explicit hierarchical structure, and compound-object substructures may be explicitly addressed for access or manipulation of the structural relationships.

The Hypertext Markup Language used to represent most web documents is a structured document format. Elements of HTML documents are explicitly tagged, and structure can be inferred from the tags. Not all substructures may be explicitly addressed—only anchor elements and embedded objects.

Fragment identifiers in URIs allow compound-object substructures to be explicitly addressed.

Other hierarchical structures are possible, but not yet supported. One facility that is notably lacking from Web implementations is transclusion—the ability to include one text object inside another by reference. For example, to include an excerpt of one document inside another, or to build a document out of section and subsection objects.

Even if transclusion were supported, compound text documents would probably be prohibitively expensive: the overhead for a transaction in the current version of the HTTP protocol is very high; hence the protocol is inefficient for retrieving a number of small objects in succession. This inefficiency is being addressed in efforts to revise the protocol.

Because HTML is essential to interoperability, it is restricted to document structures that are universally applicable. In many situations, a custom document type would support more expressive collaboration. Widespread support for custom SGML document types would enable such collaboration.

SGML is not the only structured document technology available. The multipart media types support explicit hierarchical structure, and support for them is being deployed.

The Web architecture clearly supports the requirement for structured documents. But the deployed software provides limited support. The Web is predominately used in a "publish and browse" fashion; data transmitted across the Web is largely throw-away data that looks good but has little structure. In order to use the Web as a rich collaboration platform, much more support for structured documents will be needed.

4.3. View Control of Object's Form, Sequence, and Content

> where a structured, mixed-object document may be displayed in a window according to a flexible choice of viewing options—especially by selective level clipping (outline for viewing), but also by filtering on content, by truncation or some algorithmic view that provides a more useful view of structure and/or object content (including new sequences or groupings of objects that actually reside in other documents). Editing on structure or object content from such special views would be allowed whenever appropriate.

View control can be achieved on the Web by custom processing by the information provider. A number of views can be provided, and the consumer can express their choice via links or HTML forms. For example, gateways to database query systems and fulltext search systems are commonplace. Another technique is to provide multiple views of an SGML document repository[dynatext, I4I].

Another approach to view control is client-side processing: after the document is transmitted, the reader's software could filter, sort, or truncate the data. About the only such control in wide deployment is the ability to turn off embedded images. Outline views with folding have been proposed [FOLD], but the cost of transmitting text that isn't displayed has presented a barrier.

Stylesheets are a mechanism for presenting the same data in different forms, using fonts, colors, and space to give visual structure. Support for stylesheets is an important ongoing development.

But in some systems[LINCKS, dynatext], stylesheets are much more than that: they control form, sequence, and content as Engelbart described. For example, in LINCKS, the same document can be presented and edited in abstract form, outline form, or full form depending on which stylesheet (generic presentation descriptor, in their jargon) is in effect.

At the extreme end of the spectrum, stylesheets give way to arbitrary programs that display data and interact with the reader. In a distributed system, running arbitrary programs can have dangerous consequences. But advances such as Safe-Tcl and Java make this technique of "active objects" feasible.

It's clear that the Web architecture supports custom view control. It remains to be seen whether some view control mechanisms are sufficiently valuable to be standardized.

4.4. The Basic “Hyperdocument”

> where embedded objects called “links” can point to any arbitrary object within the document, or within another document in a specified domain of documents—and the link can be actuated by a user or an automatic process to “go see what is at the other end,” or “bring the other-end object to this location,” or “execute the process identified a the other end.” (These executable processes may control peripheral devices such as CD ROM, video-disk players, etc.)

This requirement is clearly met. The hyperdocument as described above is the epitome of the Web page.

The only exception is that Engelbart refers to links as “special objects,” whereas in the Web, links are not addressable “first-class” objects—not in implementations, and not in the architecture.

4.5. Hyperdocument “Back-Link” Capability

> when reading a hyperdocument online, a worker can utilize information about links from other objects within this or other hyperdocuments that point to this hyperdocument—or to designated objects or passages of interest in this hyperdocument.

The design of the Web trades link consistency guarantees for global scalability. Links are one way , and the reverse link is not guaranteed to exist.

Aside from the intractability of maintaining global link consistency, another barrier to a distributed back-link service is privacy. Some links are sensitive, and their owners do not want them easily discovered.

These barriers do not, however, prevent Web users from utilizing back-link information. Some Web server tools [FrontPage]) maintain back-link information for the local site. And the HTTP protocol includes a mechanism—the `Referer:` field —that allows information providers to gather back-link information for their site.

Finally, there are search services that traverse the web building full-text search indexes. Some [Altavista] take advantage of the links they encounter to offer back-link services.

4.6. The Hyperdocument “Library System”

> where hyperdocuments can be submitted to a library-like service that catalogs them and guarantees access when referenced by its catalog number, or “jumped to” with an appropriate link. Links within newly submitted hyperdocuments can cite any passages

> within any of the prior documents, and the back-link service lets the online reader of a document detect and "go examine" any passage of a subsequent document that has a link citing that passage.

There are no guarantees of access on the web today. A few commercial web service providers [NaviService] guarantee server availability on a 24x7 basis, but this doesn't guarantee access to the entire global user base—any network interruption between the reader and the provider can prevent access.

In a distributed system, absolute guarantees are impossible. But reliability can be made arbitrarily good by investing in redundancy. A number of strategies for caching and replication (including Harvest) are being explored, standardized, and deployed.

Catalog numbering systems have not matured either. This is known as "the URN problem." A number of promising proposals have been made [PATH, STANF], but none is widely deployed.

Digital Libraries is an active field of research. ARPA is funding research, and the Online Computer Library Center is conducting experiments.

4.7. Hyperdocument Mail

> where an integrated, general-purpose mail service enables a hyperdocument of any size to be mailed. Any embedded links are also faithfully transmitted—and any recipient can then follow those links to their designated targets in other mail items, in common-access files, or in "library" items.

Internet Mail is possibly the world's most widely deployed information system. MIME, the Multipurpose Internet Mail Extensions, standardizes facilities for attachments and compound documents, among other things.

Though nearly every new mail system supports MIME, the installed base of pre-MIME mail systems is still significant —a majority by many estimates.

User interfaces that integrate email (and USENET news) into the Web user experience are anticipated, but not deployed.

4.8. Personal Signature Encryption

> where a user can affix his personal signature to a document, or a specified segment within the document, using the private signature key. Users can verify that the signature is authentic and that no bit of the signed document or document segment has been altered since it was signed.

There are a number of digital signature standards, but none has been globally adopted and deployed on the web.

One barrier is patent licensing. A critical feature of web technology that led to its rapid deployment was its royalty-free copyright status. Patents on public-key cryptography prevent digital signature technology from being deployed without license negotiations.

Another barrier is export control legislation. Implementations of cryptographic techniques such as encryption are considered munitions by many governments, and there are strict controls on the export of such technologies.

But the largest barrier is the social and educational one. Digital signature techniques will have to be tested in production use, and users will have to be educated about the related issues before commerce can depend on this technology.

4.9. Access Control

> Hyperdocuments in personal, group, and library files can have access restrictions down to the object level.

The distributed nature of the Web allows information providers to implement any access control policy they choose, down to the object level.

Minimal support for username/password authentication is widely deployed. This allows information providers to implement access control based on users and groups of users. But this basic facility is not robust in the face of concerted attack.

A number of mechanisms for strong authentication and confidentiality, as well as billing and payment are being standardized. A complete discussion of these mechanisms is beyond the scope of this document.

4.10. Link Addresses That Are Readable and Interpretable by Humans

> one of the "viewing options" for displaying/printing a link object should provide a human-readable description of the "address path" leading to the cited object; AND, that the human must be able to read the path description, interpret it, and follow it (find the destination "by hand" so to speak).

Document addresses in the web are designed so that they can be transcribed—written on envelopes, recited over the phone, etc. Each URI scheme has an associated public specification of how to interpret and follow its path description.

By and large, URIs are sensible to those familiar with the conventions—`http://www.ford.com` is the address of Ford Motor Company, for example.

But portions of URIs are allowed to be opaque by design—they may be pointers into an index, checksums, dates, etc.

4.11. Every Object Addressable

> in principal, every object that someone might validly want/need to cite should have an unambiguous address (capable of being portrayed in a manner as to be human readable and interpretable). (E.g. not acceptable to be unable to link to an object within a "frame" or "card.")

Every object on the web is addressable. But not every substructure within objects that someone might need to cite has a standard addressing mechanism. For example, individual pages in a postscript document, lines in a text file, pixels in an image.

These structures are, in principle, addressable. Only a standard syntax for URI fragment identifiers to address them is lacking.

In HTML documents, elements can be named and addressed. But there is no mechanism to address unnamed elements. For parties that do not have write access to a document, this presents a problem.

One solution would be to allow elements to be addressed by their structural position. There are a number of standard technologies for addressing elements of SGML documents (and hence HTML documents): TEI pointers, HyTime location ladders, and DSSSL queries. Any of these could be incorporated into web software.

Another possibility is to address strings within a document by pattern matching. One annotation system[BRIO] uses patricia trees for stable pointers into documents.

4.12. Hard-Copy Print Options to Show Address of Objects and Address Specification of Links

> so that, besides online workers being able to follow a link-citation path (manually, or via an automatic link jump), people working with associated hard copy can read and interpret the link-citation, and follow the indicated path to the cited object in the designated hard-copy document.
>
> Also, suppose that a hard-copy worker wants to have a link to a given object established in the online file. By visual inspection of the hard copy, he should be able to determine a valid address path to that object and for instance hand-write an appropriate link specification for later online entry, or dictate it over a phone to a colleague

Most of the installed base of web client software allows users to view link address. But ironically, that option is not available for printing in many cases. It would be a straightforward enhancement, well within the bounds of the existing architecture.

5. Summary

In the following table, each row represents one of Engelbart's requirements. The columns are as follows:

Architecture Support
: Is the requirement explicitly addressed in the Web architecture? Does the architecture passively allow the requirement to be met by applications? Or does the architecture conflict with the requirement?

Standard Facilities
: Are there standardized facilities that meet the requirement? Partially meet the requirement?

Ubiquitous Facilities
: Some facilities are standardized, but not required of all systems, or not supported by all installations. What facilities are assumed to be supported by all participants in the web?

Local/Proprietary Facilities

Has the requirement been met within the architecture by local applications or with the use of proprietary facilities?

Each cell contains an evaluation of whether the requirement is met (YES, NO, PASSive, or PARTial) followed by a list of relevant facilities. Missing facilities are marked with *.

Evaluation

Requirement	Architecture Support	Standard Facilities	Ubiquitous Facilities	Local/Proprietary Facilities
1. Mixed Object Documents	YES: format negotiation, typed links	PART: URI, HTML, IMG, INSERT, MIME link types*	PART: GIF in HTML	YES: JPEG in HTML, Java/Safe-Tcl in HTML, OLE, OpenDoc, Fresco
2. Explicitly Structured Documents	YES: fragment identifiers	YES: HTML, SGML, MIME	PART: HTML	YES: Panorama, OLE, LINCKS
3. View Control of Object's Form, Sequence, and Content	PASS	PART: HTTP, CGI	NO	YES: DynaWeb, Java/Safe-Tcl, Style Sheets
4. The Basic "Hyperdocument"	YES	YES: URI, HTML	YES: URI, HTML	
5. Hyperdocument "Back-Link" Capability	PASS	PART: Referer	NO	YES: local link map, back-link service
6. The Hyperdocument "Library System"	PASS	NO	NO	NO
7. Hyperdocument Mail	PASS	YES: MIME	YES: MIME	
8. Personal Signature Encryption	PASS	NO	NO	YES: S-HTTP, PEM, S/MIME, PGP, PKCS-7
9. Access Control	YES	PART: Basic auth	PART: Basic Auth	YES: MD5, SSL, S-HTTP, PGP, smart cards, etc.
10. Link Addresses That Are Readable and Interpretable by Humans	PART: URI	PART: URI	PART: URI	
11. Every Object Addressable	objects: YES substructures: PASS	PART: URI	PART: URI	
12. Hard-Copy Print Options to Show Address of Objects and Address Specification of Links	PASS	NO	NO	YES: HTML2LaTeX

6. Conclusions

Support for Engelbart's requirements is far from ubiquitous. But the architecture in no way prevents them from being realized, and the quantity of resources integrated into the system provides ample motivation for research and development.

In each area where facilities to meet the requirement are not ubiquitous, a demonstration of sufficient facilities has taken place.

This gives confidence that the requirements will eventually be met and become infrastructure.

If in fact Engelbart's requirements are an effective way to measure the viability of a platform for electronic commerce, the Web is very likely to be a viable platform for some time to come.

7. Acknowledgements

This paper has been in development since I met Douglas Engelbart and Tim Berners-Lee on the same day at Hypertext '91.

Since then, a number of collaborators have influenced my perspective on web design: Roy

Fielding, Larry Masinter, Terry Allen, and Stu Wiebel to name a few.

Thanks to Jim Miller for his encouragement and review comments.

I would also like to thank Ravi Kalakota, who invited me to write this paper and present it at [UT95].