EXHIBIT M

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Examiner: Caldwell, A. T.
DOYLE et al.	Art Unit: 2154
Application No.: 10/217,955	Response
Filed: August 9, 2002	
For: DISTRIBUTED HYPERMEDIA METHOD AND SYSTEM FOR AUTOMATICALLY INVOKING EXTERNAL APPLICATION PROVIDING INTERACTION AND DISPLAY OF EMBEDDED OBJECTS WITHIN A HYPERMEDIA DOCUMENT	

Commissioner for Patents
Alexandria, VA 22313-1450

Sir:

In response to the Office Action mailed 09/09/2004, please consider the following remarks:

Amendments to the Claims begin on page 2 of this paper.

Remarks/Conclusion begin on page 5 of this paper.

AMENDMENTS TO THE CLAIMS:

Please cancel claim 2.

This listing of the claims replaces all prior versions, and listings, of claims in the application.

1. (Original) A computer program product for use in a system having at least one client workstation and one network server coupled to a network environment, wherein said network environment is a distributed hypermedia environment, wherein said client workstation utilizes a browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies, either directly or indirectly, the location of at least a portion of said object, wherein said portion is external to said first distributed hypermedia document, wherein said object has type information associated with it utilized to identify and locate computer readable program code external to the first distributed hypermedia document, and wherein said embed text format is parsed by said browser to automatically invoke said computer readable program code, the computer program product comprising:

a computer usable medium having computer readable program code physically embodied therein, said computer program product further comprising:

computer readable program code, identified by said type information, for being automatically invoked by the browser application to cause the client workstation to display an object and enable interactive processing of said object within the display area created at said first location within the portion of the first distributed hypermedia document being displayed in the first browser controlled window.

2. (Cancelled)

3. (Original) A computer program product for use in a system having at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment, the computer program product comprising:

a computer usable medium having computer readable program code physically embodied therein, said computer program product further comprising:

computer readable program code for causing said client workstation to execute a browser application to parse a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and to respond to predetermined text formats to initiate computer instruction sequences specified by said text formats;

computer readable program code for causing said client workstation to utilize said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies, either directly or indirectly, the location of at least a portion of an object external to the first distributed hypermedia document, wherein said object has type information associated with it utilized by said browser, or by some other program, to identify and locate a sequence of computer instructions external to the first distributed hypermedia document, and wherein said embed text format is parsed by said browser to automatically invoke said sequence of computer instructions to execute on said client workstation in order to display said object and enable interactive processing of said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.

REMARKS

Claims 1-3 have been reexamined, claim 2 is canceled, and claims 1 and 3 are now pending in the application. Reexamination and reconsideration of all outstanding rejections and objections is requested.

Claims 1 through 3 are rejected under 35 U.S.C. §103(a) as being unpatentable over the admitted prior art in the U.S. Patent No. 5,838,906 ('906 patent), the teachings of Berners-Lee, Raggett I, and Raggett II, and the newly cited teaching of Toye.

Claim 2 is rejected under 35 U.S.C. §101 for same invention double patenting. Claims 1 and 3 are rejected under the judicially created doctrine of obviousness-type double patenting. Claim 2 has been canceled to obviate the double patenting rejection of claim 2 and a terminal disclaimer is attached hereto to obviate the same invention obviousness-type double patenting rejection of claims 1 and 3.

Introduction

Included with this response are a Rule 132 Declaration by Professor Edward W. Felten, Professor of Computer Science at Princeton University ("Felten II, signed October 6, 2004"), traversing the rejections of claims 1 and 6 of U.S. Patent No. 5,838,906 ("the '906 patent) based on the same references cited in this Office Action and the Rule 132 Declaration by Professor Felten submitted with the response filed May 10, 2004 ("Felten I, signed May 7, 2004). Although these declarations were prepared in response to Office Actions mailed in connection with the reexamination of the parent patent application, A/N 08/324,443 (now the 906 patent), the obviousness issues raised in those Office Actions are identical to the obviousness issues raised in the present Office Action. References to these declarations relevant to identical issues raised in the present office action will be made in the following arguments.

It is Applicants' position that the reference referred to below as Raggett II is not a publication according to 35 U.S.C. §102. However, for the purposes of the following arguments this reference is being treated as if it is prior art.

Outline of the Non-Obviousness Argument for Claims 1 and 3

A. The Claimed Invention

B. Description of the References
1. Applicants' Admitted Prior Art (Mosaic), Berners-Lee, Raggett I, and Raggett II
2. Toye

C. The Examiner's Reasoning

D. Traverse

PART I. The establishment of a *prima facie* case of obviousness requires that all the claim limitations must be taught or suggested by the prior art. MPEP §2143.03
None of the references of the proposed combination, when considered either individually or collectively, teach or suggest the claimed features of the Applicants' invention. Accordingly, a *prima facie* case of obviousness has not been established.

a. There is no suggestion or teaching in either Toye, the admitted prior art (Mosaic), Berners-Lee, Raggett I or Raggett II of automatically invoking an external application to execute on a client computer, when an embed text format is parsed, to display and interactively control an object in a display window in a hypermedia document, received over a network from a network server, being displayed in a browser-controlled window on the client computer.

b. There is no suggestion or teaching in either Toye, the admitted prior art (Mosaic), Berners-Lee, Raggett I or Raggett II of parsing an embed text format at a first location in the hypermedia document and displaying the object and enabling interactive processing of the object within a display area created at the first location within the portion of the hypermedia document being displayed.

c. Because the claim limitations are not taught or suggested by the cited references, the combination proposed in the rejection would not include the limitations of claims 1 and 3.

PART II The establishment of a *prima facie* case of obviousness requires that the claimed combination cannot change the principle of operation of the primary reference or render the reference inoperable for its intended purpose. MPEP §2143.01. The proposed combination of Toye with the combination of Mosaic, Berners-Lee, Raggett I and II would change the operation of the latter combination and render it inoperable for its intended purpose. Accordingly, a *prima facie* case of obviousness has not been established.

a. The combination proposed in the Office Action contradicts a fundamental principle of operation of the Mosaic, Berners-Lee, Raggett I and II combination requiring that the images, rendered when the Raggett embed tag is parsed, be static images.

b. The combination proposed in the Office Action would change the Mosaic, Berners-Lee, Raggett I and II combination from being a distributed system, which is a basic principle of its operation and an intended purpose.

c. The combination proposed in the Office Action would change the Mosaic, Berners-Lee, Raggett I and II combination from a system intended to give the document author control over the user's browsing experience to a system which causes the document author to lose that control.

PART III The obviousness rejection is based on a false premise and therefore reaches a false conclusion.

a. Toye does not disclose a distributed hypermedia system in which a hypermedia browser allows a user to interactively process an object embedded within a distributed hypermedia document.

b. There is no teaching in Toye of a dynamic object that would make obvious modifying the static image taught by the combination of the admitted prior art (Mosaic), Berners-Lee, and Raggett I and II into a dynamic image.

PART IV There is no motivation or teaching in the cited references to combine the references to make the claimed invention obvious.

a. The language in Toye regarding "openness and flexibility" cited by the examiner teaches away from a combination that would make the claims obvious.

b. The fundamental problems solved by the Mosaic, Berners-Lee, Raggett I and II systems (HTML browser) and the Toye system teach away from a combination that would make the claimed invention obvious.

c. It is required to consider the references in their entireties, i.e., including those portions that would argue against obviousness. *Panduit Corp. v. Dennison Manufacturing Company*, 227 USPQ 337, 345 (CAFC 1985).

DETAILED ARGUMENT

A. The Claimed Invention.

The invention, as recited for example in claim 1, is for use in a system having at least one client workstation and one network server coupled to a distributed hypermedia environment, where the client workstation utilizes a browser application, executed on the client workstation, that parses a hypermedia document to identify text formats in the document and responds to predetermined text formats to initiate processing specified by the text formats and where the browser displays a portion of a first distributed hypermedia document, received over the network from the network server, in a browser-controlled window. The hypermedia document includes an embed text format, located at a first location in the hypermedia document, that specifies the location of at least a portion of an object external to the hypermedia document. The object has associated type information utilized by the browser to identify and locate a sequence of computer instructions external to the hypermedia document.

When an embed text format is parsed by the browser, the sequence of computer instructions is automatically invoked, as a result of the parsing, to execute on the client workstation.

A computer readable medium has computer program code embodied therein for being automatically invoke by the browser application to cause the client workstation to display an object and enable interactive processing of the object within a display window created at the first location of the portion of the hypermedia document being displayed in the first browser controlled window.

The invention, as recited for example in claim 3, is for use in a system having at least one client workstation and one network server coupled to a distributed hypermedia environment.

The claim recites a browser application, executed on the client workstation, that parses a hypermedia document to identify text formats in the document and responds to predetermined text formats to initiate processing specified by the text formats.

The browser displays a portion of a first distributed hypermedia document, received over the network from the network server, in a browser-controlled window. The hypermedia document includes an embed text format, located at a first location in the hypermedia document, that specifies the location of at least a portion of an object external to the hypermedia document. The object has associated type information utilized to identify and locate an sequence of computer instructions external to the hypermedia document.

When an embed text format is parsed by the browser, the sequence of computer instructions is automatically invoked, as a result of the parsing, to execute on the client workstation.

When the automatically invoked application executes on the client workstation, the object is displayed and interactive processing of the object within a display window created at the first location of the portion of the hypermedia document being displayed is enabled.

B . Description of the References

1.a. Applicants' Admitted Prior Art

The specification of the '906 patent (Applicants' Admitted Prior Art) describes a browser application, e.g., Mosaic, that functions as a viewer to view HTML documents. There are several ways to retrieve an HTML document from a network server, all of which require user interaction with the browser. [Felten I, paragraph 8]. The browser then retrieves a selected published source HTML document from a network server by utilizing a uniform resource locator (URL) that locates the HTML document on the network and stores a temporary local copy of the HTML source document in a cache on the client workstation.

The browser application then parses the local copy of the HTML document, renders the temporary local copy of the HTML document into a Web page , and displays the rendered Web page in a browser-controlled window. [Felten I, at paragraph 21]. During the rendering step, the browser may retrieve information external to the local copy of the HTML document, such as source files referenced by IMG tags, render the images from the retrieved files as static graphic images, and insert the images into the Web page of the HTML document, for display to the user.

There is no further interaction with the source HTML document or the local copy of the source HTML document subsequent to its being rendered and displayed. If a user believes the source HTML document has changed (s)he can click a refresh button in the browser GUI which causes the browser application to retrieve the source HTML document from the network server again, store a local copy again, parse and render again the newly retrieved local copy of the source HTML document, and replace the display of the previous version of the retrieved source HTML document with the subsequently retrieved version in the browser-controlled window or another window. For example, if the source HTML document were a price list of goods the user might refresh the document to determine if the prices had changed.

Although the browser application passively displays links, from text or picture elements of a first hypermedia document to other external data objects, a user may browse by actively selecting links to retrieve information identified by a link. The retrieved information either replaces the first hypermedia document or is displayed in a separate window other than the window displaying the hypermedia document. Mosaic has the capability of allowing the user to invoke an external application to open a new window to display file types that cannot be displayed by Mosaic (helper applications).

Some browsers, such as Mosaic, include the capability of rendering images in certain formats, such as GIF , designated as a native format. These images may be placed inline in an HTML document using the IMG element, which specifies a source location, URL, of the source file to be rendered by the browser, and displayed in the rendered format of the document. All static images referenced by IMG or FIG tags specified in the HTML document must be retrieved by the browser prior to rendering the HTML document.

For data formats that can not be rendered by the browser application itself, i.e., data in a foreign or non-native format such as ".TIF," Mosaic launches helper applications, in response to a user's command, in a separate window to view certain types of file types. As described in the specification, the mechanism for specifying and locating a linked object is an HTML anchor "element" that includes an object address in the format of Uniform Resource Locator (URL).

Many viewers exist that handle various file formats such as TIF. When a user commands the browser program to invoke a viewer program (helper application), typically by clicking on an anchor with a mouse, the viewer is launched as a separate program. The viewer program displays the image in a separate "window" (in a windowing environment) or on a separate screen. This means that the browser program is no longer active while the viewer program is active. The viewer program is completely independent of the browser after being invoked by the browser so that there is no communication between the viewer program and the browser program after the viewer program has been launched.

As a result, the viewer program continues to run, even after the browser program execution is stopped, unless the user explicitly stops the viewer program's execution.

Mosaic was a significant advance that made the WWW easily accessible and gave Web page authors a powerful tool to provide simplified user-activated access to viewing of hypermedia documents and related external data objects anywhere on the WWW network.

There is no disclosure of automatically invoking an external application to enable interactive processing of an object in a display area of a hypermedia document being displayed by the browser .

1.B Berners-Lee (Berners-Lee, T., et al., Hypertext Markup Language (HTML), Internet Draft, IETF, pages 1-40, (June 1993)

The Berners-Lee reference is a specification for the HTML markup language. HTML is a language used by Web page authors to describe the structure and desired contents of their pages. A browser parses an HTML document to determine its structure and then displays the specified items as a rendered Web page within a browser window.

This reference describes a model in which Web pages are written by a Web page author, then distributed by a Web server to a browser, and viewed as a Web page displayed in the browser window by the browser's user. The user views a page, and then clicks a hyperlink or button, or enters some text, to select another page to view.

There is no disclosure in the reference relating to building a browser or how a browser works, nor is there disclosure in the reference of automatically invoking an external application to enable interactive processing of an object in a display area of a hypermedia document being displayed by the browser.

1.c. Raggett I (Raggett, D., HTML+(Hypertext Markup Language), (July 23, 1993))

Raggett I is a document entitled "HTML+ (Hypertext Markup Language) A proposed standard for a light weight presentation independent delivery format for <u>browsing</u> and <u>querying</u> information across the internet" [emphasis added]. In pertinent part, Raggett I generally relates to allowing Web page authors to display static images of equations and simple drawings in a Web page. At page 3, describing the HTML+ Document Format, it is stated that "HTML+ departs slightly from pure presentation independence by allowing Web page authors to specify rendering hints to give Web page authors greater control over the final appearance of documents."

At pages 4 and 5, Inlined Graphics or Icons are discussed. It is stated that these elements are treated like characters in the text and an example of the IMG tag is given:

This line has a egyptian hieroglyph at the end of the
line. <img src = "ankh.tiff">

It is further stated that the URL notation is used to name the source of the graphics data and that sophisticated HTML+ editors should allow Web page authors to modify images using an external editor. It is also stated that larger inlined images should be specified with the FIG tag.

At page 6, Raggett's proposed EMBED tag is described that provides a simple form of object level embedding that is very convenient for mathematical equations and simple drawings. Raggett's proposed EMBED tag would allow Web page authors to continue to use familiar standards, such as *TeX* and *eqn*. It is also stated that images and complex drawings are better specified by using the FIG or IMG elements.

Raggett's proposed EMBED tag would utilize a type attribute to specify a MIME content type to be used by a browser to identify a rendering application, such as a shared library or external filter, used to render embedded data. An example of rendering the embedded data is given as returning a pixmap which is a data structure holding a static image.

An example of Raggett's proposed EMBED tag is given as follows:

<embed type="application/eqn">2 pi int sin(omega t)dt</embed>

In this example the embedded data is "2 pi int sin(omega t)dt" and the type information is "application/eqn". In this example, the embedded data is processed by the *eqn* application to render a static graphic image of the embedded data in the following form:

$$2\pi \int \sin(\omega t) dt$$

The reference also states that sophisticated browsers can link to external editor applications for creating and revising embedded data.

It is also stated at page 12 that when using the FIG tag, instead of using a *src* attribute, an EMBED element can be included immediately following the <FIG> tag and that this is useful for simple graphs etc. defined in an external format.

At page 13 the *ismap* attribute of the FIG tag is described. It is stated that arbitrary areas of the figure can be designated as hypertext links.

There is no disclosure in the reference relating to building a browser or how a browser works, nor is there disclosure in the reference of automatically invoking an external application to enable interactive processing of an object in a display area of a hypermedia document being displayed by the browser.

1.d. Raggett II (Raggett, D., Posting of Dave Raggett, dsr@hplb.hpt.hp.com to www-talk@nxocOl.cern.ch (W-WWW-TALK public mailing list) (Posted June 14, 1993))

The position of the Applicants is that Raggett II is not a publication complying with 35 U.S.C. §102. However, in the following it will be assumed that Raggett II is prior art.

Raggett II is an email message from David Raggett to Torben Nielsen and Bill Janssen having the subject line "HTML+ support for eqn & Postscript".

This reference quotes an email from Nielsen stating that he has lots of documents he wants to put on the Web and that without support for equations it is quite difficult. It also quotes an email from Janssen stating he would like to send encapsulated Postscript in his documents.

The email then states that the HTML+ DTD makes both these requests possible by providing the capability to embed foreign data inline in the HTML source. The document then gives an example of Raggett's proposed EMBED tag and states that the browser identifies the format of the embedded data from the "type" attribute. It is also stated that building in support for a large number of formats has the danger of leading to very large programs for browsers and that this can be avoided by using a common API for rendering foreign formats, e.g., as rendering functions that take a sequence of bytes and return a pixmap.

It is then stated that browsers can then be upgraded to display new formats by binding MIME content types to the function names for those formats and that the functions could be implemented as separate programs driven via pipes and stdin/stdout or as dynamically linked libraries (DLLs). It is also stated that foreign data can be put in a separate file referenced by a URL.

There is no disclosure in the reference relating to building a browser or how a browser works, nor is there disclosure in the reference of automatically invoking an external application to enable interactive processing of an object in a display area of a hypermedia document being displayed by the browser.

2. Toye, G., et al., SHARE: A methodology and Environment for Collaborative Product Development, Proceedings, Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1993, IEEE, pp. 33-47, April 22, 1993.

Toye is a paper describing a SHARE project that seeks to apply information technologies in helping design teams gather, organize, re-access, and communicate both informal and formal design information to establish a "shared understanding" of the design and design process. The paper also presents research and strategies undertaken to build an infrastructure toward the realization of SHARE. Two components of the SHARE environment are NoteMail and DIS (Distributed Information Services).

Fig. 5, at page 39, depicts an application-oriented view of the SHARE architecture. The top level architecture of SHARE is a set of services communicating over the Internet. Some of these services include DIS, link managers, and constraint managers. The diagram illustrates that SHARE can communicate over the Internet, as can other information services such as the World Wide Web, Databases, Catalogs, and Libraries.

In the SHARE architecture email is the primary medium for both human communication and tool integration. For example, NoteMail messages are formatted in MIME (Multi-purpose Internet Mail Extension) that enables them to be sent as ordinary e-mail and read using any MIME-compliant mail reader.

The shaded tear drop in Fig. 5 shows that the SHARE environment consists of three classes of tools. One class is NoteMail and DIS which helps engineers capture and manage file information. NoteMail is a tool for collaborative editing (i.e., editing by several members of a team) of engineering documents within an engineering team.

NoteMail messages are formatted in MIME (Multi-purpose Internet Mail Extension), the Internet standard for multimedia mail, and can be sent as ordinary e-mail and read by using any MIME-compliant mail reader. NoteMail uses a "Format" data type that captures and preserves the spatial arrangement of information items on each NoteMail page.

It is stated that an interesting feature of NoteMail is the open architecture of its viewer. Unlike most other engineering notebooks and multimedia authoring environments, any application that displays through an X-server can insert its output (audio, video, or graphics) dynamically into a notebook page through a "dynamic window".

This is accomplished in two steps. First, after a data object or file is selected by a user for inclusion in the notebook the system will invoke the appropriate application for display in the notebook. Subsequently selecting the displayed data with a mouse will restart the original application so that data can be edited or updated without leaving the network environment. It is stated that this functionality is similar to opening a file using Macintosh finder and automatically invoking the appropriate application for processing that file.

It is then stated that other engineering notebooks lack this openness and flexibility and only allow processing of a handful of input formats.

Because NoteMail messages are to be sent by email, full copies of the messages are not sent to everyone. Instead it is more efficient to store the components of the message in one place and just transmit a set of reference pointers. NoteMail uses an object-oriented knowledge base, known as DIS, for this repository.

There is no disclosure in the reference of building a hypermedia browser, as that term is used in claims 1 and 3, of modifying applications that edit or update files or objects, nor is there disclosure in the reference of automatically invoking an external application to enable interactive processing of an object in a display area of a hypermedia document being displayed by the browser.

C. THE EXAMINER'S REASONING

The examiner states that the combination of Applicant's admitted prior art in view of Berners-Lee, Raggett I and Raggett II does not explicitly teach a method that "enables interactive processing of said object". The combination teaches a method that embeds static objects, as opposed to dynamic objects, within distributed hypermedia documents.

It is then stated that Toye, on the other hand, discloses a distributed hypermedia system in which a hypermedia browser allows a user to interactively process an object embedded within a distributed hypermedia document, citing Toye's description of NoteMail, particularly p. 40, col. 2, first complete paragraph.

It is then concluded that it would have been readily apparent to a skilled artisan to modify the method discussed above, combining the teachings of the admitted prior art in view of Berners-Lee, Raggett 1 and Raggett II, by further modifying the combination's static embedded object to be a dynamic embedded object as taught by Toye. It is stated that the modification would be apparent based on Toye's teaching that its architecture provides openness and flexibility.

D. TRAVERSE

This rejection is respectfully traversed for the following reasons.

The entire Felten II declaration is incorporated herein as an independent traverse of the rejection of claims 1 and 3. The following argument recapitulates parts of the traverse set forth in Felten II, with citations to relevant parts thereof, and presents additional arguments not present in Felten II. Further, the argument also includes citations to Felten I.

The basic requirements of a Prima Facie Case of Obviousness are set forth in MPEP §2143:

> To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations.
>
> The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in applicant's disclosure. In re Vaeck, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).]

The level of skill in the relevant art is set forth in the Felten I declaration as:

> The benchmark for a person having ordinary skill in the art (PHOSA) is a person who is just graduating from a good computer science program at a college or a university, not a star student but just a typical, average student, or a person who has gained equivalent knowledge in the industry. This person knows how to do things in conventional ways but does not exhibit an unusual level of innovative thinking. [Felten I, paragraph 15].

PART I. The establishment of a *prima facie* case of obviousness requires that all the claim limitations must be taught or suggested by the prior art. MPEP §2143.03
None of the references of the proposed combination, when considered either individually or collectively, teach or suggest the claimed features of the Applicants' invention. Accordingly, a *prima facie* case of obviousness has not been established.

a. There is no suggestion or teaching in either Toye, the admitted prior art (Mosaic), Berners-Lee, Raggett I or Raggett II of automatically invoking an external application to execute on a client computer, when an embed text format is parsed, to display and interactively control an object in a display window in a hypermedia document, received over a network from a network server, being displayed in a browser-controlled window on the client computer.

As described above, in the Mosaic, Berners-Lee, Raggett I and II system a hypermedia document retrieved by the browser is rendered into a set of ordered static presentation formats that are subsequently displayed by the browser. As described in Berners-Lee and Raggett I and II, browsers have the ability to render graphics files into static images that can be inserted inline into the set of static presentation formats to be subsequently displayed by the browser.

Raggett I and II teach the use of an external rendering application invoked by the browser to process a graphics file in a foreign format (a format not handled by the browser itself) and to return a static image that the browser inserts inline in the static presentation form of the document that is subsequently displayed by the browser.

Accordingly, as acknowledged by the examiner, the Mosaic, Berners-Lee, Raggett I and II combination teaches that the browser displays a static, non-interactive image and the claimed feature of automatically invoking an external application to execute on the client computer to interactively control an object displayed in a display window in the hypermedia document is not taught or suggested by that combination of references.

Further, as set forth in Felten I, the rendering applications invoked in the Mosaic, Berners-Lee, Raggett I and II combination return a static image and terminate. The browser inserts the static image returned by the rendering application into the set of static presentation formats comprising the presentation form of the hypermedia document, prior to the document being displayed by the browser. Thus, the types of rendering applications taught by Raggett I and II that are invoked when Raggett's EMBED tag is parsed are not capable of providing interactive processing of an object displayed within a display area created in the hypermedia document <u>being displayed</u> in the browser controlled window as required by claim 6. Instead, the Raggett rendering applications terminate <u>before</u> the hypermedia document is displayed by the browser.

Toye discloses a NoteMail viewer that allows a user to view a static image of a notebook page. The first full paragraph on page 40 of Toye describes an authoring environment and a viewing environment.

When authoring a NoteMail page the author may actively <u>select</u> a data file or object to be included in the NoteMail page and then a static image of the file is displayed in the NoteMail page. [Felten II, at paragraphs 33-35]. The image displayed in the page must be static because

Toye states that subsequently selecting the data with a mouse will restart the original application so that the data can be edited or updated. [Toye at page 40, first full paragraph]. The fact that the original application must be restarted to interact with the data displayed in a NoteMail page teaches that the displayed data was static and that no interaction with the data was possible prior to its selection with a mouse. [Felten II, at paragraph 35].

Toye states that when an object or file is selected by the user the system will automatically invoke the application for display in a NoteMail page. Further, Toye teaches that the application launching functionality is similar to opening a file using Macintosh Finder. [Toye at page 40, first full paragraph]. Thus, Toye teaches that automatic invoking is a result of user selection, not parsing as required by claims 1 and 3, and that the result of the user's interactive selection is similar to opening a file using Macintosh Finder, where the application launched processes the file in its own window. [Felten II, at paragraph 36].

Accordingly, Toye teaches away from automatic invocation of an external application when a document is parsed to enable interactive processing of the object but instead teaches that an object must be selected by a mouse to invoke an application to enable interactive processing.

Thus, like the Mosaic, Berners-Lee, Raggett I and II combination, a static presentation format of the NoteMail page is displayed by the viewer. Subsequently selecting a static image displayed in the NoteMail page launches an application that allows a user to edit or update the data.

b. There is no suggestion or teaching in either Toye, the admitted prior art (Mosaic), Berners-Lee, Raggett I or Raggett II of parsing an embed text format at a first location in the hypermedia document and displaying the object and enabling interactive processing of the object within a display area created at the first location within the portion of the hypermedia document being displayed.

In the admitted prior art (Mosaic) and Berners-Lee combination a hypermedia document selected by the user is located on the Internet, retrieved by the browser, rendered into an ordered set of static presentation formats by the browser, and subsequently displayed in a browser controlled window.

The modification to the browser suggested by Raggett I and II does not change this fundamental viewing paradigm. As described above, the static images returned by external applications invoked in the Raggett system are inserted in line by the browser into the ordered set of static presentation formats comprising the displayable form of the hypermedia document. In Raggett I and II, the Raggett EMBED tag located at a first location in the hypermedia document is parsed, a rendering application is invoked that returns a static image and terminates, the static image is inserted at the first location in the set of static presentation formats, and the presentation form of the document is then displayed by the browser. Since the rendering application has terminated before the set of static presentation formats is displayed by the browser, it is fundamentally incapable of providing interactive processing of an object being displayed in the display area of a hypermedia document being displayed in the browser controlled window.

Turning next to the Toye reference, NoteMail messages are formatted in MIME (Multi-purpose Internet Mail Extension) and a new "Format" MIME data type is defined, for NoteMail to capture and preserve the spatial arrangement of information on a NoteMail page. The MIME "Format" data is stored separate from the text portions of the document [Felten II, at paragraph 31]. There is no teaching in NoteMail of using text formats, within the document text, intended to initiate processes specified by those text formats. Further, there is no teaching in NoteMail of parsing an embed text format at a first location and displaying and enabling interactive processing within the first location because, in NoteMail, the location of information is specified elsewhere, by the "Format" data type.

Additionally, the Toye reference teaches that any application that displays through an X-server can be restarted by subsequently selecting the displayed data in a NoteMail page with a mouse, so that the data can be updated or edited. There is no teaching of modifying an application to allow interactive processing within a display area of a hypermedia document being displayed in a browser controlled window. Toye teaches that any application able to display through an X-server would allow editing and updating of a file in a window controlled by the editing application. Toye provides no teaching that new applications should be created to provide this editing capability. Rather, he teaches that any existing application which is capable of being displayed through an X-server is suitable for this purpose. As Professor Felten points out [Felten II, at paragraph 38], this teaches away from the proposed combination, since existing editor applications at the time of the claimed invention were designed to be run in their own windows, under their own control, and contained menu bars and other graphical interface elements which would interfere with the editors being useable if run so as to provide interaction in a display area in a first location of the document being displayed.

Further, Toye teaches that the application launching functionality is similar to opening a file using Macintosh Finder. [Toye at page 40, first full paragraph]. In Macintosh Finder opening a file launches an application in a separate window. [Felten II, at paragraph 34]. Thus, Toye teaches away from enabling interaction in a display area in a first location of a document being displayed.

c. Because the claim limitations are not taught or suggested by the cited references, the combination proposed in the rejection would not include the limitations of claims 1 through 3.

As set forth below, the references provide no motivation for the combination proposed by the rejection, and such a combination would change the basic operating principles of the Mosaic, Berners-Lee, Raggett I and II Web browser technology. However, even if the combination were possible it would not include the limitations of the pending claims. [Felten II, at paragraphs 46-51].

Such a combination would not automatically invoke an external application to enable interactive processing within a display area of a hypermedia document being displayed by the browser because the Mosaic, Berners-Lee, Raggett I and II combination teaches that external data is rendered to a static bit map and then displayed by the browser, and Toye teaches that external data is displayed as a static bit map that must be selected by a mouse to launch an editor application in a separate window. [Felten II, at paragraph 47].

Instead, the combination, if it could be constructed, would include the Raggett method of creating a static bitmap within a browser window in such a way that a user clicking on that static bitmap would launch an editor program in an external window, as in Toye. [Felten II, at paragraph 50].

This combination would not show automatic invocation of the editor program when the hypermedia document is parsed or enable interactive processing within a portion of the first hypermedia document being displayed in the browser window, as required by claims 1 and 3. Instead, the external editor application of Toye would be invoked only if the user took the additional manual action of selecting the static image by clicking on it, causing interactive processing to be enabled in an external window when the external application was restarted. [Felten II, at paragraphs 48-50].

Even if the Toye combination were to show interactive processing within a portion of the first hypermedia document being displayed in the browser window, the combination would still not show automatic invocation of the editor program when the hypermedia document is parsed, as required by claims 1 and 3.

Thus at least two elements of claims 1 and 3 would be missing from the proposed combination. [Felten II, at paragraph 51].

PART II The establishment of a *prima facie* case of obviousness requires that the claimed combination cannot change the principle of operation of the primary reference or render the reference inoperable for its intended purpose. MPEP §2143.01. The proposed combination of Toye with the combination of Mosaic, Berners-Lee, Raggett I and II would change the operation of the latter combination and render it inoperable for its intended purpose. Accordingly, a *prima facie* case of obviousness has not been established.

a. The combination proposed in the Office Action contradicts a fundamental principle of operation of the Mosaic, Berners-Lee, Raggett I and II combination, requiring that the images, rendered when the Raggett embed tag is parsed, be static images.

Raggett I teaches uses of Raggett's proposed EMBED tag that require the returned image to be static. At page 12 of Raggett I, it is stated that, instead of using the *src* element, Raggett's proposed EMBED element can be used as an element of the FIG tag. It is known in the art that the FIG element is utilized to display static images in the displayed version of the HTML document. Since the use of Raggett's proposed EMBED tag, as a substitute for a *src*-defined static image file in this context is not qualified, Raggett's proposed EMBED tag is required to return only a static image, or it would cause the FIG tag to function incorrectly. [Felten I, at paragraph 44].

The requirement that Raggett's proposed EMBED tag return only a static image is further reinforced by the discussion in Raggett I of active areas at page 13. The *ismap* attribute described with respect to the FIG tag causes the browser to send mouse clicks on a figure back to the server using a selected coordinate scheme. Arbitrary areas of the figure can be designated as hypertext links. The Web page author thus creates a semantic correspondence between areas of the figure and Web pages that can be retrieved by clicking over these various areas. If the figure displayed were to be interactively changed then this semantic correspondence would be destroyed. Further, a mouse click can have only a single function. Since the *ismap* feature causes the browser to send mouse clicks to the server, the mouse click can not be utilized to interact with the image and the image must be static. Thus, an explicitly stated intended purpose of the Mosaic, Berners-Lee, Raggett I and II system is to allow the image returned by the Raggett EMBED-tag rendering application to be compatible with the *ismap* attribute of the FIG tag. [Felten II, at paragraph 19].

Thus, the ability to use Raggett's proposed EMBED tag, instead of the *src* attribute, within the FIG tag requires that a static and non-interactive image be returned. If this were not the case then Raggett I would require special discussion on the use of Raggett's proposed EMBED tag as an attribute within the FIG tag. No such discussion is included and thus Raggett I teaches that the image returned by Raggett's proposed EMBED tag must be static and non-interactive.

Accordingly, the reasoning of the rejection, that it would have been obvious to modify the static image taught by the Mosaic, Berners-Lee, Raggett I and II combination to be a dynamic object as taught by Toye, is a direct contradiction of the teaching of Raggett I and would change the principle of operation of the Mosaic, Berners-Lee, Raggett 1 and II combination, and render it inoperable for one of its intended purposes. If the displayed static image of the Mosaic, Berners-Lee, Raggett I and II combination were modified to be dynamic as suggested by the rejection, then the intended purpose of allowing the image returned by the Raggett rendering function to be compatible with the *ismap* attribute of the FIG tag would be rendered inoperable.

b. The combination proposed in the Office Action would change the Mosaic, Berners-Lee, Raggett I and II combination from being a distributed system, which is a basic principle of its operation and an intended purpose.

The admitted prior art describes a hypertext document as "a document that allows a user to view a text document displayed on a display device connected to the user's computer and to access, retrieve and view other data objects that are linked to hypertext words or phrases in the hypertext document. In a hypertext document, the user may "click on," or select, certain words or phrases in the text that specify a link to other documents, or data objects." [Application at page 2, line 6-7] "A hypermedia document is similar to a hypertext document, except that the user is able to click on images, sound icons, video icons, etc., that link to other objects of various media types, such as additional graphics, sound, video, text, or hypermedia or hypertext documents." [Application at page 2, lines 22-25]. When the hypermedia document is displayed on a browser program the browser responds to the selection of a link to retrieve and display the hypermedia document or data object referenced by the link.

A distributed hypermedia system is a "distributed" system because data objects that are imbedded within a document may be located on many of the computer systems connected to the Internet." [Application at page 6, line 27-29].

In the Mosaic, Berners-Lee, Raggett I and II combination the Web-page author specifies the location of a linked-to object in tags such as A (anchor), IMG, or FIG defined by the HTML mark-up standard. Thus the author is responsible for and has control of the location of referenced objects. Since the Mosaic and Berners-Lee combination teaches a distributed system, the objects may be located on any computer connected to the Internet.

Further, the browser retrieves a copy of a source document from its server location, renders the presentation form of the document, and displays the document. The original source document cannot be edited or updated by a browser user.

Thus, the Mosaic, Berners-Lee, Raggett I and II combination teaches a model in which static pages can be published by anyone, on a server anywhere in the world, and read by anyone. The pages are connected by simple, unidirectional links that are used only to navigate from one page to another. A page is created and edited by its author, using a separate editing application, and is viewed, but not modified, by its readers using a separate browser application. [Felten II, at paragraph 12].

Accordingly, the Mosaic, Berners-Lee, Raggett I and II combination was designed to operate as a distributed system where objects may be stored anywhere on the Internet and retrieved by utilizing a browser application, by simply clicking on a link in a document displayed by the browser, to access another document located anywhere on the Internet.

In contrast, Toye teaches a system for collaborative editing of engineering documents within an engineering team, using a single object-oriented database (DIS) to store documents.

Toye teaches the use of a centralized, object-oriented database for storage of the workgroup's documents.

> Multimedia engineering documents containing raw text, encoded images, audio clips, video clips, etc. can get quite large. Sending such documents via email to everyone on a large design team can be costly in terms of both time and storage. Instead of transferring full copies to everyone, it is more efficient to store the components of the message in one place and just transmit a set of reference pointers. NoteMail uses an object-oriented knowledge base, known as DIS, for this repository function.
>
> Conceptually, DIS provides a centralized information storage and management service for all the data associated with a design: CAD files, e-mail messages, specifications, simulation results, and so forth. In practice, most data remains physically under the control of the application that created it; a persistent object is created in DIS to serve as a reference pointer or "handle."
> [Toye at p. 40-41, emphasis added].

The use of a centralized, object-oriented database makes sense given the goal of Toye to support collaboration within an engineering workgroup. [Felten II, at paragraphs 21-24]. Further, links between objects are created in the centralized database and not in the NoteMail page. [Toye, page 41 at the first partial paragraph].

The rejection states that it would have been obvious to modify the static combination taught by the Mosaic, Berners-Lee, Raggett I and II combination to be a dynamic object as taught by Toye.

However, any attempt to combine the centralized storage of referenced objects taught by Toye with the Mosaic, Berners-Lee, Raggett I and II combination would change the basic principle of operation of the combination being modified. A fundamental principle of operation and an intended purpose of the Mosaic, Berners-Lee, Raggett I and II combination is to provide a distributed system that allows objects to be stored anywhere on the Internet. A combination with Toye would turn that distributed system into a centralized database system, thereby destroying its distributed nature. Such a fundamental change teaches away from any combination of the Mosaic, Berners-Lee, Raggett I and II distributed system and the Toye centralized system.

Thus, the Mosaic, Berners-Lee, Raggett I and II and Toye references would not make the combination of claims 1 and/or 3 obvious to the PHOSA, because the differences in the basic principles under which the Mosaic, Berners-Lee, Raggett I and II combination and the Toye system operate, with regard to the storage and referencing of objects from a displayed page, are fundamentally different and incompatible.

c. The combination proposed in the Office Action would change the Mosaic, Berners-Lee, Raggett I and II combination from a system intended to give the document author control over the user's browsing experience to a system which causes the document author to lose that control.

The Web model of the Mosaic, Berners-Lee, Raggett I and II combination teaches a system which is based upon a publish-once/view-many paradigm. In that model, a document author is able to create an HTML document file which specifies precise locations for the various data objects that the browser will render for display in the page that the user sees. [Felten II, at paragraphs 13-14]. The combination proposed in the Office Action would be contrary to this basic principle.

The Web model insures document integrity. The document author can be assured that the fully-rendered document that (s)he originally created is going to appear the same for every user who subsequently retrieves that document for viewing. The end user, on the other hand, can be assured that the document being viewed has not been changed since it was last edited by the document author. This is extremely important in any document publishing system, since publishing systems are, by nature, intended to allow end users to rely on the published form of documents as accurate representations of the author's intended vision.

This notion of assuring data integrity is a fundamental principle of the Web model. That principle of data integrity assurance is destroyed in the proposed combination with the teachings of Toye. The Toye reference teaches a collaborative editing environment where any user can modify the data objects which are then rendered for display in the document. [Felten II, at paragraph 21]. An example of this is seen where Toye states: "for example, recipients can redo analyses and simulations with their own parameters" [Toye at page 40, first column, second full paragraph under the Notemail heading]. Once a recipient redoes such an analysis with different parameters than the original author specified, the resultant data object to be displayed in the document changes. When a subsequent user views the document, (s)he sees not the original document in the form specified by the original creator of the document, but rather the rendered document reflecting the sum total of changes made to both the document and the rendered data objects by any and all users who have accessed and modified that document since its creation.

Since any user can change the data objects in the Toye system, no user can rely on the document as a reflection of the original author's vision. An unavoidable consequence of the combination of Mosaic, Berners-Lee, Raggett I and II with Toye, therefore, would be a publishing system where the information communicated by the published document could be modified by users over time to the point where it would bear no resemblance to the document which the author intended to publish. This would render the Web unsuitable for its intended purpose.

Another important principle of the Web model taught by the Mosaic, Berners-Lee, Raggett I and II combination is that of referential integrity. In the Web model, the HTML document author can specify the specific locations, contained in "hypertext links," from which the browser will retrieve new HTML documents when users click upon those links. These links are easily specified by the document author, since they are directly specified through embed text formats in the document text. In the Web model, these are simple unidirectional links which are used only to navigate from document to document. The document author explicitly defines these links, and they are resolved and acted upon directly by the browser application. [Felten II, at paragraph 15].

This simple and lightweight linking model allows for the design of efficient distributed hypermedia browser applications which are optimized for the viewing of documents comprising both text and distributed data objects. It also allows for rapid navigation by the user from document to document, without limitations imposed by the physical location of either the document text or the data objects to be displayed. [Felten II, at paragraph 23].

Since it is primarily a publishing and information retrieval system, the Web system taught by the Mosaic, Berners-Lee, Raggett I and II combination employs unidirectional links, which can only be defined by the author, to insure that only the author's vision of the navigational paths out of the document is reflected in the final document that users can retrieve. This provides referential integrity to the system, which is a basic principle of the Web's fundamental design.

Since HTML authors can rely on the referential integrity of the documents they create, large information systems can be created via collections of multitudes of inter-linked distributed hypermedia documents. Without the enforcement of this referential integrity, the Web model would become unsuitable for the creation of such information systems.

Toye teaches that links should be bi-directional, and that they should be managed by separate applications. Toye teaches that links within the Share system can communicate changes in both directions. A consequence of this is that, in the Toye system, the definition of a link can be changed by agents out of the control of the document author. As Professor Felten explains: "The bi-directional links of Toye can, for example, represent formal constraints that connect two documents, so that a change in either of the two documents causes a corresponding change to happen automatically in the other document. This model is appropriate within an engineering workgroup, but it doesn't make sense on the Web, where hyperlinks often link documents written by different people who may not know or trust each other. For example, on the Web, I can create a page that links to the CNN home page; but it would not be appropriate for me to create a Toye-style link that would allow me, by changing my page, to cause changes on CNN's home page. Instead, Web hyperlinks follow a more appropriate (for the Web's goals) model in which only I can modify my own page, and only CNN can modify their page. This difference teaches away from the use of a Web browser with Toye." [Felten II, at paragraph 30]

In the Toye system, therefore, the document author can no longer be assured that the functionality of any link within an authored document will always be what the document's creator intended. This makes sense in a system designed for team-based collaborative editing of inter-linked documents, where one would naturally desire to have changes made by any collaborator instantly reflected for all to see, and for those changes to propagate through series of linked documents. In the proposed combination of Toye with Mosaic, Berners-Lee, Raggett I and II, however, the assurance of referential integrity that is so vital to the usefulness of the Web model would be unavoidably destroyed.

Furthermore, as has been discussed above, the combination with Toye would result in an embedded graphic presentation format of data that, while it would be static at the time of viewing, could change over time as various users would modify the corresponding data object, as enabled by Toye's collaborative editing environment. As a result the *ismap* functionality of the FIG tag of Raggett II would be rendered unusable, since the various intra-image links defined by the FIG tag would lose their semantic correspondence, and therefore their referential integrity, as originally defined by the HTML document's author.

So it is clear, therefore, that such a combination with Toye would destroy both the data integrity and the referential integrity that are fundamental principles behind the design of the prior art Web system, and that the proposed combination would therefore render the Web model unsuitable for its intended purposes.

PART III The obviousness rejection is based on a false premise and therefore reaches a false conclusion.

a. Toye does not disclose a distributed hypermedia system in which a hypermedia browser allows a user to interactively process an object embedded within a distributed hypermedia document.

The Office Action, at page 7, lines 4-7, states that Toye discloses a distributed hypermedia system in which a hypermedia browser allows a user to interactively process an object embedded within a distributed hypermedia document. However, this statement is incorrect in view of the precise meaning of the various terms defined in the Mosaic, Berners-Lee, Raggett I and II combination.

The admitted prior art describes a hypertext document as "a document that allows a user to view a text document displayed on a display device connected to the user's computer and to access, retrieve and view other data objects that are linked to hypertext words or phrases in the hypertext document. In a hypertext document, the user may "click on," or select, certain words or phrases in the text that specify a link to other documents, or data objects." [Application at page 2, line 3-6] "A hypermedia document is similar to a hypertext document, except that the user is able to click on images, sound icons, video icons, etc., that link to other objects of various media types, such as additional graphics, sound, video, text, or hypermedia or hypertext documents." [Application at page 2, line 22-26]. When the hypermedia document is displayed on a browser program the browser responds to the selection of a link to retrieve and display the hypermedia document or data object referenced by the link.

A distributed hypermedia system "is a "distributed" system because data objects that are imbedded within a document may be located on many of the computer systems connected to the Internet." [Application at page 6, lines 27-29].

The use of HTML allows the Internet to be an open system where a standard protocol is implemented by each computer connected to the internet. The structure of the document is defined by the author utilizing particular sets of characters that have a universal meaning.

In contrast, Toye teaches a system that is not a distributed system but requires that all referenced objects be stored in a single data base called DIS. [Toye, page 40, column 2, first and second paragraphs below the heading "Distributed Information Service (DIS)].

The NoteMail pages described in Toye use DIS as the central repository for referenced objects in contrast to the ability of a distributed hypermedia document to reference objects located in computers at different geographic locations. Thus, the Toye system does not teach or suggest using distributed hypermedia documents and its principle of operation is incompatible with the use of distributed hypermedia documents. [Felten II, at paragraph 24].

Also, for the same reasons Toye does not teach the use of a "distributed hypermedia environment" as that term is defined in the admitted prior art and used in claims 1 and 3. The use of the centralized storage of referenced objects is crucial to the intended purpose of the Toye system and contradicts the basic requirements of a distributed hypermedia environment. [Felten II, at paragraph 25].

Toye does not teach a hypermedia browser application, as that term is defined in the admitted prior art, Berners-Lee, and Raggett I and II, understood by the PHOSA at the time the application was filed, and as used in claims 1 and 3. Toye teaches no software application that parses distributed hypermedia documents or that uses text formats, and it does not teach other browser-related elements of the pending claims, such as parsing of distributed hypermedia documents by a browser, identifying text formats in distributed hypermedia documents and responding to predetermined text formats to initiate processing specified by those formats, utilizing a browser to display at least a portion of a distributed hypermedia document in a browser-controlled window, and parsing an embed text format in such a document. [Felten II, at paragraphs 26-27].

Further, the Toye reference teaches that information can be organized by adding links between objects where the links themselves are objects stored in the DIS database. [Toye, page 41, col. 1, first partial paragraph]. Thus, Toye is not a hypermedia system because, in the admitted prior art, Berners-Lee, and Raggett I and II combination, links are defined by the author as text formats in the hypermedia document and resolved by the browser application.

The Mosaic, Berners-Lee, Raggett I and II combination teaches the use of a hypermedia document that is a text document where some characters within the text are interpreted as mark-up tags specified by the HTML standard. The mark-up "tags" give structure to the document. [Berners-Lee, page 5, Felten II, at paragraph 14].

In contrast, Toye teaches that the structure, i.e., spatial arrangement of information in a NoteMail page, is preserved by a non-standard MIME "Format" data type defined by the Toye authors for the specific NoteMail system being described. [Toye, page 40, first column, last partial paragraph, Felten II, at paragraph 31]. Accordingly, Toye does not teach the use of a hypermedia document, in the sense of the Mosaic, Berners-Lee, Raggett I and II combination, or the embedding of an object in such a hypermedia document. NoteMail pages are therefore not analogous to Web-style hypermedia documents.

Also, there is no teaching in Toye of interactively processing an object embedded in a hypermedia document. Toye teaches that data displayed in a NoteMail page must be selected via a mouse click by the user to restart an application in order to update and edit data. The type of application described in Toye is any application that displays through an X-server. [Toye page 40, second column, first full paragraph]. There is no teaching of modifying such an application to process an object embedded in a hypermedia document. Further, Toye teaches that most data remains physically under the control of the application that created it, suggesting that the data must be processed using the normal interface for the application. [Felten II, at paragraphs 36-37].

b. There is no teaching in Toye of a dynamic object that would make obvious modifying the static image taught by the combination of the admitted prior art (Mosaic), Berners-Lee, and Raggett I and II into a dynamic image.

In view of the above, there is no teaching in Toye that would make the modification proposed in the rejection apparent to the skilled artisan. The failure of Toye to suggest or teach a distributed hypermedia system or the use of an analogous hypermedia document, as well as the other fundamental incompatibilities in architecture described above, would teach away from attempting to combine any features of the Mosaic, Berners-Lee, Raggett I and II combination with the Toye system.

The rejection implies that Toye teaches a dynamic object that meets the limitations set forth in claims 1 and 3. The term "dynamic object" is not used in the pending claims. However, as set forth above, the "dynamic object" described in Toye is an object that can only be activated by clicking on a static image displayed in a NoteMail page. The link between the "dynamic object" and an application to process the "dynamic object" is stored in an external database, not the NoteMail page itself. Thus, the external application for processing the "dynamic object" is not automatically invoked when an embed text format within the document is parsed nor is interactive processing of an object displayed in a display window of a hypermedia document enabled.

Accordingly, there is no teaching or suggestion in Toye of modifying the Mosaic, Berners-Lee, Raggett I and II system to make claims 1 and 3 obvious.

PART IV There is no motivation or teaching in the cited references to combine the references to make the claimed invention obvious.

a. The language in Toye regarding "openness and flexibility" cited by the examiner teaches away from a combination that would make the claims obvious.

The rejection states that the modification of the static object in the Mosaic, Berners-Lee, Raggett I and II system would have been apparent based on Toye's teaching that its architecture provides openness and flexibility.

However, the quoted language in Toye is doing nothing more than describing the benefits of the NoteMail system editor compared to other engineering notebook projects, specifically referring to the NoteMail editor's ability to insert data in different formats into a NoteMail page. As described above, Toye teaches that any application that displays using an X-server can insert a static image of a file into a NoteMail page when the file is selected by the NoteMail author. [Felten II, at paragraphs 40-41]. There is no suggestion there that NoteMail could or should be combined with any other system. Thus, the quoted language teaches away from modifying the NoteMail editor since it is already superior to the other known engineering notebook projects.

Additionally, the level of skill in the art cannot be relied upon to provide the suggestion to combine references. MPEP §2143.01 (quoting *Al-Site Corp. v. VSI Int'l Inc.*, 50 USPQ2d 1161 (Fed.Cir. 1999). In the rejection, the general and nebulous Toye language regarding "openness and flexibility" is not related to any possible motivation to combine the references. [Felten II, at paragraph 41]. It is merely highlighting advantages of the NoteMail system over other editors commonly used for engineering collaboration systems. In fact, even if it were proper to rely on the skill in the art to provide a motivation to combine, a PHOSA would only find among these references the strong suggestion that they are not combinable.

b. The fundamentally different problems solved by the Mosaic, Berners-Lee, Raggett I and II systems (HTML browser) and the Toye system teach away from a combination that would make the claimed invention obvious.

A possible source for a motivation to combine references is the nature of the problem to be solved. MPEP 2143.01. Here, the Mosaic, Berners-Lee, Raggett I and II combination and the Toye system solve problems of a completely different nature and have structures and implementations that are fundamentally incompatible.

A list of some of the fundamental differences between the teachings of the Mosaic, Berners-Lee, Raggett I and II and the Toye reference is given in Felten II:

> Toye teaches collaborative editing of documents; Berners-Lee teaches that documents are created by an author and read (without editing) by a set of readers. Toye teaches storage of documents in a centralized object-oriented database; Berners-Lee teaches that documents can be retrieved from anywhere and everywhere on the

> Internet. Toye teaches that display structure is specified using a separate "Format" data type, outside a text document; Berners-Lee teaches that display structure is specified by markup commands within a text document. Toye teaches rich, bi-directional links implemented by separate applications; Berners-Lee teaches simple unidirectional links, providing only navigation and implemented by a browser. Toye teaches that users need not know where documents are located; Berners-Lee teaches that users know URLs, which contain location information. [Felten II, at paragraph 42].

The nature of the problem solved by the Mosaic, Berners-Lee, Raggett I and II system is the need to allow authors to publish and distribute widely on the Internet documents that can be retrieved and easily viewed by end users, without regard to the types of hardware or operating systems utilized by the computers connected to the Internet. [Application at page 1, line 16-30].

To solve this problem, the Mosaic, Berners-Lee, Raggett I and II references teach a model in which static pages can be published by anyone, on a server anywhere in the world, and read by anyone with a connection to the Internet. The pages are connected by simple, unidirectional links that are used only to navigate from one page to another. A page is edited by its author using a separate editor application, and is viewed, but not modified, by its readers using a separate browser application. [Felten II, at paragraph 12].

The nature of the problem solved by the Toye reference is the need to create a system for collaborative editing of engineering documents within an engineering team. [Toye at page 36, topic 5].

To solve this problem, Toye teaches using a single object-oriented database to store the documents needed by an engineering workgroup, [Felten II, at paragraph 24] where the data base includes bi-directional links between objects. [Felten II, at paragraph 29-30].

Because of the fundamentally different problems solved by the references, the disparate techniques and structures utilized in one system are not relevant or useful in the other. For example, the use of the collaborative editing techniques of Toye would be contrary to the publish-and-view philosophy of the Internet, as embodied in the Mosaic, Berners-Lee, Raggett I and II combination. Further, the centralized storage technique of Toye works well for highly structured engineering design, but is contrary to the distributed nature of the Mosaic, Berners-Lee, Raggett I and II combination.

c. It is required to consider the references in their entireties, i.e., including those portions that would argue against obviousness. *Panduit Corp. v. Dennison Manufacturing Company*, 227 USPQ 337, 345 (CAFC 1985).

The Toye reference, when considered in its entirety, teaches a Method and Environment for Collaborative Product Management [Toye Title] to apply information technologies to help design teams gather, organize, re-access, and communicate both informal and formal design information to establish a "shared understanding" of the design process. [Toye Abstract]. Toye teaches email as the primary medium for both human communication and tool integration. [Toye at page 39]. The SHARE environment is depicted in Fig. 4 on page 38 and depicts a number of Powerbook computers connected by email to a File Server that provides shared access to files. [Toye page 38]. The information to be shared in the collaborative group is stored in a central data base called DIS (Distributed Information Services) that helps engineers work as a team to capture, organize, retrieve, modify and share design knowledge without their having to know details such as file formats and locations.

Significantly, web browsers of the type taught by the Mosaic, Berners-Lee, Raggett I and II combination existed at the time of publication of Toye but the designers of the SHARE project chose not to use them. [Felten II, at paragraph 26-28]. Such a decision made sense because the goal of the SHARE project, to allow collaborative editing and centralized, highly-structured data management, is inconsistent with the goals of the open, distributed hypermedia model taught by the Mosaic, Berners-Lee, Raggett I and II combination.

Thus, the designers of the SHARE project, innovative engineers who recognized that realizing their vision, even in the relatively circumscribed world of engineering, would be a massive undertaking [Toye, at page 46, first column, last paragraph] did not attempt to modify or redesign the web browser taught by the Mosaic, Berners-Lee, Raggett I and II combination. Instead they designed the NoteMail system which is centralized, not distributed, and which does not use hypermedia documents as that term is used in claims 1 and 3.

The level of skill in the art is:

> The benchmark for a person having ordinary skill in the art (PHOSA) is a person who is just graduating from a good computer science program at a college or a university, not a star student but just a typical, average student, or a person who has gained equivalent knowledge in the industry. This person knows how to do things in conventional ways but does not exhibit an unusual level of innovative thinking. [Felten I, at paragraph 15].

The PHOSA does things in a conventional way. The entire Toye reference teaches a collaborative environment that is the result of a massive undertaking by innovative engineers and professors. [Toye at page 46, first column, last paragraph]. As is discussed extensively above, NoteMail teaches a model of information sharing and organization that does not use the Web browser paradigm of the Mosaic, Berners-Lee, Raggett I and II combination, but instead uses an architecture fundamentally incompatible with the Web. Thus the Toye reference teaches away from modifying the Mosaic, Berners-Lee, Raggett I and II combination as proposed by the rejection. [Felten II, at paragraph 32].

CONCLUSION

In view of the foregoing, Applicants believe all claims now pending in this Application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested.

If the Examiner believes a telephone conference would expedite prosecution of this Application, please telephone the undersigned at (925) 944-3320.

Respectfully submitted,

[illegible]

Charles E. Krueger
Reg. No. 30,077

LAW OFFICE OF CHARLES E. KRUEGER
P.O.Box 5607
Walnut Creek, CA 94596
Tel: (925) 944-3320 / Fax: (925) 944-3363