# 831 PH Ex. 13

Application of: Doyle et al.

Application Num.: 90/006,831

Filed: October 30, 2003

For: Distributed Hypermedia Method for Automatically Invoking External Application Providing Interaction and Display of Embedded Objects Within a Hypermedia Document

Examiner: Caldwell, A.T.

Art unit: 2157

# Declaration of Edward W. Felten

I, Edward W. Felten, declare as follows:

1. I have been retained by Eolas and the Regents of the University of California to serve as an expert in the field of computer science and Internet software.
2. I filed a previous declaration in this matter. That declaration recites my technical expertise and attaches my Curriculum Vitae. I hereby incorporate my previous declaration into this declaration by reference.

## I. Introduction

3. I have been asked to address the arguments presented in the Office Action mailed August 17, 2004 ("the Office Action") in connection with the reexamination of United States Patent No. 5,838,906 ("the '906 patent") that the claims of the '906 patent are unpatentable as being "obvious". For the reasons described in this declaration, I disagree with the arguments presented in the Office Action and, instead, believe that the claims of the '906 patent fully meet the requirements for patentability over the cited references, as those patentability arguments have been described to me.
4. To familiarize myself with the issues involved in the rejection of the claims, I have reviewed numerous documents, including the following: the '906 Patent and its file history, the documents sited in the previous office action mailed on March 12, 2004, the documents cited in the present Office Action, and all other documents referenced or cited in this declaration.
5. My previous declaration presented background material on the early history of Web technology and the prior art to the '906 Patent.

## II. Response to the Unpatentability Arguments Raised in the Office Action

6. My previous declaration described my understanding of the legal standard for obviousness, the level of ordinary skill in the art at the time of the '906 Patent's invention, and the nature and purpose of the invention disclosed in the '906 Patent. I hereby incorporate that discussion into this declaration by reference.

### A. The Grounds of Rejection

7. Claims 1 and 6 of the'906 patent have been rejected by the United States Patent Office as being obvious under 35 U.S.C. Sec. 103(a); as being unpatentable over the admitted prior art in the '906 patent and teaching of Berners-Lee, Raggett I, Raggett II, and Toye.

8. The Office Action asserts that a combination of the Berners-Lee, Raggett I, Raggett II, and Toye references would embody the relevant claims of the '906 Patent. For the reasons described below, I find this assertion to be incorrect.

9. From my knowledge of the field, my own personal experience, and the state of the art in 1994, to the extent that a Person Having Ordinary Skill in the Art ("PHOSA") was familiar with the teachings of the art cited in the Office Action, I find that the rejection of claims 1 and 6 as obvious is incorrect.

### B. What Berners-Lee and the Raggett References Teach a PHOSA

10. In my previous declaration, I discussed the teachings of these references. I incorporate that discussion here by reference.

11. The previous Office Action proposed a combination of these three references. However, that proposed combination would lack the claim element of automatically invoking an executable application to enable interactive processing, as acknowledged in the latest Office Action.

> The combination of patentee'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.

(Office Action at p. 6)

12. The Berners-Lee reference 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 edited by its author using a separate editor application, and is viewed, but not modified, by its readers using a separate browser application.

13. Berners-Lee teaches that the browser renders a page, translating it into a set of fixed static images to be displayed, before the page is displayed to the user.

14. Berners-Lee teaches that the structure of a document is specified by markup commands that are interspersed within the text of the document. (See, e.g., Berners-Lee at p. 5.)

15. Berners-Lee teaches that the browser parses the text of a document in order to render that document, and that the browser handles the detection and resolution of hyperlinks.

16. The model taught by Berners-Lee is well suited for the purpose of Berners-Lee, which is to create a worldwide system for viewing and navigating static, published documents on a wide variety of client computers.

17. The Raggett references are directed to the problem of increasing the number of static data formats that can be viewed by users of the Berners-Lee system. Accordingly, Raggett teaches, consistently with Berners-Lee, that data is to be rendered into a static image before it is displayed.

18. Raggett teaches the use of an external "filter" program that is used to render data that is encoded in a format the browser cannot understand. This filter program does what the browser would do: it takes a description of what to display, and generates from it a fixed image to be painted onto the screen. Raggett teaches that this filter program finishes executing before the image it generates is painted onto the screen.

19. Raggett teaches that the rendered image produced by the external filter should not be interactive. This teaching can be seen, for example, in the description of the FIG and ISMAP features in Raggett I, as discussed in paragraph 44 of my previous declaration. Raggett I teaches that its EMBED tag can be placed within a FIG element:

> Instead of the *src* attribute, you can include an EMBED element immediately following the <fig> tag. This is useful for simple graphs, etc. defined in an external format.

(Raggett I at p. 12, emphasis in original) When the FIG element is used in conjunction with the ISMAP parameter (as described in the "Active areas" section of Raggett I, p. 13), the FIG element's display area becomes an image map: any mouse clicks made by the user within the visual depiction of the embedded data will be interpreted by the browser as pertaining to the image-map feature, and will therefore be intercepted by the browser and sent by the browser to the web server. In order to do this, the browser must intercept mouse clicks within the depiction of the embedded data, and this can only happen if that depiction does not itself provide interactivity.

20. Berners-Lee, Raggett I and Raggett II, alone or in combination, do not teach the claim element of enabling interactive processing of an object. Indeed, they teach away from the provision of interactive processing within the boundaries of a web page.

## C. What Toye Teaches a PHOSA

21. Toye is directed to the creation of a system for collaborative editing (a term that would be understood by a PHOSA to refer to editing of one or more documents by multiple people) of engineering documents within an engineering team, using a single object-oriented database to store the documents needed by an engineering workgroup.

22. The Office Action characterizes Toye as follows:

> 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. **(See Toye: p. 40 description of NoteMail, particularly p. 40, col. 2, first complete paragraph)**.

(Office Action at 6:23-26, emphasis in original)

23. Contrary to the Office Action's assertion, Toye does not teach the use of a "distributed hypermedia document," as that term is used in the '906 claims. The term's meaning, as understood by a PHOSA, was reiterated in the '906 Patent's specification. For example:

> A distributed hypertext or hypermedia document typically has many links within it that specify many different data objects located in computers at different geographic locations connected by a network.

('906 Patent at 2:59-62)

24. Rather than teaching that the documents accessed by a user could be "located at computers at different geographic locations," Toye teaches the use of a single centralized, object-oriented database for storage of a 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. However, it contradicts the Office Action's assertion that Toye uses the distributed hypermedia documents of the '906 claims. Indeed, by teaching centralized storage, Toye teaches away from the use of distributed hypermedia documents.

25. For the same reason, Toye does not teach the use of a "distributed hypermedia environment," as that term is used in the '906 claims. The environment provided by Toye is not "distributed" in the sense of the '906 claims, since it relies on the centralization of a user's document storage in one place. Toye teaches away from the use of a distributed hypermedia environment.

26. Likewise, Toye does not teach the use of a hypermedia browser, as that term is used in the '906 claims. Toye teaches no software application that parses distributed hypermedia documents, and it does not teach other browser-related elements of the '906 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.

27. Toye does use the term "hypermedia browser" but with a different meaning. For example, the "hypermedia browser" of the '906 claims must parse hyperlinks from within a text document, but Toye does not provide that feature. See also the other deficiencies of Toye described in the previous paragraph.

28. At the time of Toye, a PHOSA would have known about web browser technology, and would have known that Web browser applications were available to run on widely used platforms. Yet Toye teaches that a new document viewing application (NoteMail) should be developed, rather than using existingWeb browser technology. This clearly teaches away from the use of Web browser technology with Toye.

29. Rather than teaching the use of standard hyperlinks (as described, e.g., in the '906 Patent at 2:37-47), Toye teaches the use of rich, bi-directional links (i.e., links traversable in both directions) between objects.

> The information can also be organized by adding links between objects. The links are themselves first-class objects that can be annotated with semantic labels and constraints characterizing the nature of their dependency. For example, some links may simply be hypertext pointers, used to organize e-mail message into discussion threads and link them to related documents and data. Others may be used to represent a [sic] formal constraints in a behavioral model. In either case, maintaining the links is the job of external applications that provide navigation, constraint management, change notification and other services. These applications can attach daemons to the links, which are run automatically when either side changes.

(Toye at p. 41, first partial paragraph) These links provide functionality far beyond the simple hyperlinks used on the web, and they are implemented by separate applications. Again, this teaches away from the use of the hypermedia browser of Berners-Lee.

30. 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.

31. Unlike Berners-Lee, Toye teaches that the structure of multimedia content is not specified within the text of the main or enclosing text document, but is specified elsewhere, for example in a separate MIME-part that uses Toye's "Format" data type. (See, e.g., Toye at p. 40, bottom of first column.)

32. The Toye reference teaches an entire system, of which the NoteMail module cited in the Office Action is just one part. The system taught by Toye has different aims, and teaches a different model, than Berners-Lee and Raggett.

### D. What Toye Teaches About Interaction with External Programs

33. Toye teaches that NoteMail interacts with an external program by first displaying a static snapshot of the external content. If the user clicks on that static snapshot, the external editor application is restarted in a separate window.

34. As noted in the Office Action (at 7:3-6), the key to understanding Toye's interaction with external programs can be found in Toye's discussion of restarting the external editor when the user clicks on the snapshot of the external content:

> When a data object or file is selected for inclusion in the notebook, the system will automatically invoke the appropriate application for displaying that item in the notebook.... Subsequently selecting the displayed data with a mouse will restart the original application, so that the data can be edited or updated without leaving the notebook environment. The functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file.

(Toye at p. 40, col. 2, first full paragraph) It is clear from this discussion that before the data can be edited, the user must select the displayed data with the mouse and the application must be restarted. Since the user must take specific action to select the data before editing is enabled, the editor is not "automatically invoke[d] ... in order to display said object and enable interactive processing" as required by the '906 claims.

35. The fact that the application must be restarted in order to enable editing tells us that the application could not have been running already in a way that enabled editing. (If it were, no restarting would be necessary.) Thus Toye teaches that the data, as originally displayed (i.e., before it is selected by the user) cannot be edited. It follows that all that is displayed initially is a static, non-editable snapshot of the data.

36. Toye does teach the launching of a separate application, but that application is launched in a separate window from the enclosing document. Toye teaches that its application launching "functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file." (Toye at p. 40, col. 2, first full paragraph) In the Macintosh Finder, I know by personal experience, and a PHOSA would have known, that opening a file launched an application in a separate window area. I note also that the Finder did not invoke an application automatically, but did so only in response to a mouse click selection by the user.

37. Toye does talk about displaying a document's data within the notebook environment. Even accepting, for the sake of argument, that the notebook environment of Toye is a browser, this still does not meet the requirements of the '906 claims. The '906 claims require not only that the interactivity be provided within the browser window, but that it be provided "within a display area ... within the portion of said first distributed hypermedia document being displayed ...." ('906 Patent at 17:23-28). This element is not taught by Toye. (Nor is it taught by Berners-Lee or either of the Raggett references.)

38. Indeed, Toye teaches the use of external editor programs that have not been modified from their standard versions. (See, e.g., Toye at p. 40, col. 2, first full paragraph: "any application that displays through an X-server") Such unmodified programs are not suitable for use within an enclosing document display, because the unmodified programs conventionally display menus and button bars at the top, and other graphical elements around their edges. External application windows with these elements on their borders cannot naturally be displayed within a document display; at most they could be displayed in a window area elsewhere in a windowing environment, as discussed in the previous paragraph. To enable a reasonable editing experience within a document display, the applications would have to be modified; but Toye teaches that they are not modified.

## E. No Teaching or Suggestion to Combine

39. Neither Toye nor any other reference suggests a combination of Toye with Berners-Lee, Raggett I and Raggett II.

40. Regarding a teaching or suggestion to combine, the Office Action says only this:

> 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 I, and Raggett II, by further modifying

> the combination's static embedded object to be a dynamic embedded object as taught by Toye. Such a further modification would have been apparent based on Toye's teaching that its architecture provides openness and flexibility (See Toye: p. 40 col. 2 second complete paragraph).

(Office Action at 6:28-34) (I note that the term "dynamic embedded object" does not appear in the '906 claims, and that what the Office Action calls the "dynamic embedded object as taught by Toye" is not the "object" of the '906 claims, because it is not displayed in the manner required by the '906 claims, as explained above.)

41. The Office Action is incorrect when it implies that the cited paragraph of Toye suggests a combination of Toye with a web browser. The cited paragraph of Toye reads as follows:

> We are aware of only one other multimedia editor with such an architecture, MediaMosaic [citation]. Other engineering notebook projects, by contrast, lack this openness and flexibility. For example, the Virtual Notebook System [citation] can display only static bitmaps; GE's Electronic Design Notebook [citation], which is built on FrameMaker, can run only those applications whose output formats are compatible with the handful of input formats that FrameMaker accepts.

(Toye at p. 40 col. 2 second complete paragraph) In this paragraph, Toye is simply asserting that its system has advantages over other engineering collaboration systems. Toye offers more than static bitmaps; it offers also the ability to click on those bitmaps and launch an external application (in a separate window, as discussed above). Toye offers more than just FrameMaker-compatible formats. "Openness and flexibility" are little more than buzzwords here. Nothing in this paragraph would teach a PHOSA that Toye could or should be combined with a web browser.

### F. The References Teach Away From the Suggested Combination

42. As discussed above, the cited references teach away from a combination. 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.

43. Importantly, Toye teaches away from the use of distributed hypermedia documents, which is the central idea of Berners-Lee.

### G. The Suggested Combination Would Not Embody the '906 Claims

44. The Office Action suggests a combination of Berners-Lee, Raggett I, Raggett II, and Toye:

> 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 I, and Raggett II, by further modifying the combination's static embedded object to be a dynamic embedded object as taught by Toye.

(Office Action at 6:28-31)

45. Even granting, for the sake of argument, that such a combination would be possible, the result would not embody the '906 claims.

46. The resulting system, contrary to the language of the '906 claims, would not automatically invoke an external program to enable interactive processing within a browser window.

47. The Berners-Lee/Raggett combination teaches that external data is rendered to a static bitmap that is then displayed within a browser window. Toye teaches that external data is displayed as a static image, and if the user clicks that image, an editor application is launched in a separate window.

48. The combination, assuming it could be constructed, would therefore involve the use of the Raggett method to create 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.

49. This combination would not provide automatic invocation of the editor program. The editor program would not be invoked immediately when the user visited the enclosing web page. Instead, the invocation would happen only after the user took the additional manual action of selecting the static image by clicking on it.

50. This combination would not provide interactive processing within the portion of the first hypermedia document displayed within the browser window. Interactive processing would occur only within the external editor window that was launched in response to the user's mouse click.

51. The fact that these two claim elements are missing is consistent with the fact that they are missing in all four of the references being combined.

## III. Conclusion

52. For the reasons explained above, I conclude that the rejection of claims 1 and 6 as being unpatentable is incorrect. The claims of the '906 patent would not have been obvious in view of the references cited in the Office Action.

I declare that all statements made herein of my knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both under 18 U.S.C. Section 1001, and that such willful false statements may jeopardize the validity of the patent.

Dated: October 6, 2004

[illegible]

Edward W. Felten

# 831 PH Ex. 14

Application of: Doyle et al.

Application Num.: 90/006,831

Filed: October 30, 2003

For: Distributed Hypermedia Method for Automatically Invoking External Application Providing Interaction and Display of Embedded Objects Within a Hypermedia Document

Examiner: Caldwell, A.T.

Art unit: 2157

# Declaration of Edward W. Felten

I, Edward W. Felten, declare as follows:

1. I have been retained by Eolas and the Regents of the University of California to serve as an expert in the field of computer science and Internet software. My Curriculum Vitae, which recites my technical expertise, is attached hereto to as Exhibit A.

## I. <u>Qualifications</u>

2. I graduated with Honors from the California Institute of Technology in 1985, with a B.S. degree in Physics. I received an M.S. in Computer Science in 1991, and a Ph.D. in Computer Science in 1993, both from the University of Washington.
3. I am currently a Professor of Computer Science at Princeton University, where I have taught since 1993. I was originally hired at Princeton as an Assistant Professor, in 1993. I was promoted to Associate Professor in 1999, and to Professor in 2003.
4. I am the author or co-author of numerous publications relating to computer science and Internet software. These publications are listed in my CV.
5. I have been asked to address the arguments presented in the Office Action mailed March 12, 2004 ("the Office Action") in connection with the reexamination of United States Patent No. 5,838,906 ("the '906 patent") that the claims of the '906 patent are unpatentable as being "obvious". For the reasons described in this declaration, I disagree with the arguments presented in the Office Action and, instead, believe that the claims of the

'906 patent fully meet the requirements for patentability over the cited references, as those patentability arguments have been described to me.

6. To familiarize myself with the issues involved in the rejection of the claims, I have reviewed numerous documents, including the following: the '906 Patent and its file history, the documents cited in the Office Action, and all other documents referenced or cited in this declaration.

7. Before specifically addressing the cited references and unpatentablity arguments raised in the Office Action, I believe that it is important to discuss the relevant state of the browser art as it existed in 1994. My discussion is based on my experience as a computer science researcher and teacher, and as a Web user and network software developer. From this experience, I have gained an independent understanding of how the browser art developed.

## II. <u>Relevant State of the Art in 1994</u>

8. In 1994, the Web was young, and browsers were a relatively new technology. Browsers offered only a very limited form of interactivity. A page could contain hyperlinks, on which the user could click to view another page. A page could be a form to be filled out by the user, with a "submit" button which, when clicked, caused the user to see another page.

9. Another technology, known as "helper applications," was implemented in the Mosaic browser. This technology allowed the browser to link to an external program, in cases where the browser encountered a file whose format the browser did not understand. For example, if the user clicked on a hyperlink that pointed to a file in .mpeg format (i.e., a movie in MPEG format), then the browser would launch an external MPEG-viewer program and pass the .mpeg file to that program. The result would be that the MPEG program ran, in a separate window from the browser.

10. Helper applications allowed the browser to link to an external program, but that program could not provide interactivity within the browser window. The helper application was just an external program that ran on the same computer, in a separate window.

11. None of these methods allowed a Web page author to place fully interactive objects within the confines of a Web page's display.

12. These methods are all implemented in today's browsers, and they are all in use on the Web today.

## III. Response to the Unpatentability Arguments Raised in the Office Action

13. I have been told by patent counsel for Eolas and the Regents that a patent may not be obtained, even though the invention is not anticipated, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made, to a person having ordinary skill in the art to which the subject matter pertains. I have further been told that I need to make a four step inquiry to evaluate "obviousness" in which the scope and content of the prior art are to be determined; the level of ordinary skill in the pertinent art resolved; and against this background, the obviousness or nonobviousness of the subject matter is determined. I have also been told that such secondary considerations as commercial success, long felt but unresolved needs, failure of others etc. might be utilized to give light to the circumstances surrounding the origin of the subject matter sough to be patented.

14. As a "useful general rule" I have been told that references that "teach away" cannot serve to create a meritorious case of obviousness. Also, I have been told that proceeding contrary to the accepted wisdom is strong evidence of nonobviousness. In addition, I have been told that the prior art must "suggest" or "motivate" one of ordinary skill in the art to combine the prior art to make the claimed invention and must further have taught that such a combination would have a "reasonable expectation of success".

### A. The Level of Skill In the Art

15. My benchmark for what ordinary skill in the art means 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 an average student – or a person who has gained an equivalent level of knowledge through experience in the industry. This person knows how to do things in conventional ways but does not exhibit an unusual level of innovative thinking.

16. In 1994, those of ordinary skill in the art were just becoming familiar with the Web and Web browsers. One of ordinary skill would have had a general idea of how the Mosaic browser worked, and would have been familiar with hyperlinks, forms, and helper applications.

### B. The Grounds of Rejection

17. Claims 1 and 6 of the'906 patent have been rejected by the United States Patent Office as being obvious under 35 U.S.C. Sec. 103(a); as being unpatentable over the admitted prior art in the '906 patent and teaching of Berners-Lee, Raggett I, and Raggett II. While I understand that the patent attorneys for Eolas and the Regents are challenging whether Raggett I and Raggett II are really "prior art" to the '906 patent, I have been asked to

assume for the purposes of my analysis that both the Raggett I and Ragett II references would have been "prior art".

### B. The '906 Patent

18. The claims of the '906 Patent describe a technology that allows web page authors to include, within the boundaries of a web page, interactive objects. This is done (briefly stated) by including in the web page's HTML text an embed text format, that provides information about where to get the object's data, along with information to identify and locate an executable application that will be invoked on the client computer to display the data and to provide interactivity with it, and by providing a web browser that knows how to parse the HTML to extract the embed text format, how to use type information to identify and locate the executable application, how to invoke the executable application, to execute on the client computer, and how to interface to the executable application so as to allow the user to interact with it within the boundaries of the browser window.

### C. Prior Art Browsers

19. The Office Action cites the applicants' admitted prior art. I have reviewed all prior art references referenced in the '906 Patent's file history. It appears that the Office Action's discussion of this prior art focuses on the Mosaic browser, which was the most advanced prior art browser.

20. Mosaic, and other prior art browsers, executed on a client computer, and operated by downloading copies of web pages (and other files, such as embedded static images) over a network from web servers. After downloading a copy of a file, Mosaic would sometimes keep a copy of that file in a local cache, on the user's client computer. Caching allowed the file to be referenced more quickly if it was needed again later.

21. After downloading a file, Mosaic would parse that file (i.e., analyze its structure) to determine how the file should be displayed on the screen. Mosaic would then paint the contents of the file into a browser window.

22. When Mosaic, or another prior art browser, was used to view web pages, several steps stood between the author of the web page and the user who was viewing it. First, the file would be copied, at least once and perhaps more times, while in transit between the web server and the user's browser. Second, the file would be written in one format (typically, HTML) but displayed in another form, by rendering the HTML into a visual representation that would actually be presented to the user.

23. Because these steps stood between the author and the user, there was no realistic way for the user to edit the web page on the client workstation. The user did not have access to the version of the page that was distributed – that version lived on the server, and it wouldn't make sense to let an arbitrary user edit the contents of somebody else's web page.

24. In addition, because web pages were written in one format (HTML) and viewed in another (visual representation), it did not make sense to talk about editing and viewing a document in the same window. Web page authors would typically work with two separate windows open, one (a browser) to see what the visual representation looked like, and another (an external editor) to actually modify the page's HTML representation. An author would fiddle with the HTML, then click the save button in the editor and the refresh button in the browser to see what the visual representation of the page looked like, then fiddle with the HTML some more, and so on until he was satisfied with the page's appearance.

**D. The Berners-Lee Reference**

25. 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 visual representation of the specified items within a browser window.

26. The Berners-Lee reference teaches a model in which Web pages are written by an author, then distributed by a Web server to a browser, and viewed as a static item by the browser's user. The user views a page, and then clicks a hyperlink or a button, or enters some text, to select another page to view.

27. In the model taught by Berners-Lee, a user interacts with the Web by moving from one static page to another. Thus Berners-Lee teaches away from the provision of rich interactivity within a page.

28. Berners-Lee teaches a language for authoring web pages, but it does not teach how to build a browser or how a browser works.

**D. The Raggett I Reference**

29. Raggett I suggests some modifications to the HTML system taught in Berners-Lee. The overall teaching of Raggett I is very similar to that of Berners-Lee.

30. Like Berners-Lee, Raggett I does not teach how to build a browser or how a browser works.

31. Raggett I teaches the use of the same model as Berners-Lee, in which Web pages are essentially static, and the user interacts with the Web by moving from page to page. Accordingly, Raggett I teaches away from the provision of rich interactivity within a page.

32. Raggett I is motivated by the problems of Web page authors. Authors want to be able to include in their pages information in a wide variety of formats. For preexisting content, an author wants to be able to use the content in the format in which it was originally created. For new content, an author wants to be able to choose a format well suited to a particular type of content. For example, if the content consists of mathematical

equations, the author wants to be able to be able to use a format designed for describing equations.

33. At the time of Raggett I, browsers such as Mosaic could handle only a limited set of data formats. Web page authors had noted a need for the display of static pages in more, and more varied, data formats.

34. One known method for displaying more formats was to do server-side translation. In this method, a web page author would take a document in some format, and generate a static image file from it. For example, an author might take a file describing a diagram, and generate from that file a static image, in GIF format, depicting the diagram. The web server could then deliver the GIF file to the browser, which would know how to render it within a web page.

35. Another known method to enable the display of more formats was to build support for displaying additional formats into the browser itself. Among the disadvantages of this approach were that it made the browser larger and more complicated, and that it required a new version of the entire browser to be distributed to a user before that user could view the new format.

36. Raggett I proposed a slight extension of this method, in which, rather than receiving an image, the browser receives information in some foreign format, and then uses an external program to render that information into an image, which the browser displays within the web page. This is a simple and natural extension of the browser's ability to display static images.

37. This extension is described in the following paragraph, which is also cited in the Office Action:

> The EMBED tag provides a simple form of object level embedding. This is very convenient for mathematical equations and simple drawings. It allows authors to continue to use familiar standards, such as TeX and eqn. Images and complex drawings are better specified using the FIG or IMG elements. The type attribute specifies a registered MIME content type and is used by the browser to identify the appropriate shared library or external filter to use to render the embedded data, e.g. by returning a pixmap. It should be possible to add support for new formats without having to change the browser's code, e.g. through using a common calling mechanism and name binding scheme. Sophisticated browsers can link to external editors for creating or revising embedded data. Arbitrary 8-bit data is allowed, but &, < and > must be replaced by their SGML entity definitions. For example <embed type="application/eqn">2 pi int sin (omega t) dt</embed> gives [image of equation appears here].

(Raggett I at p. 6)

38. This paragraph teaches a method for displaying new types of *static* information within a Web page. The teaching of the use of static information is evident for several reasons.

39. First, the use of static information is consistent with the teaching of the remainder of Raggett I and with the teaching of Berners-Lee that preceded it.

40. Second, Raggett I motivates its proposed embed tag by referring to two types of data that one might want to display: "mathematical equations and simple drawings". These are types of data that one would want to display statically.

41. Third, Raggett I says that Raggett's proposed embed tag "allows authors to continue to use familiar standards, such as *TeX* and *eqn*." (italics in original). These are well-known formats for describing the display of static data. TeX is used to specify the typesetting of textual documents; it is still widely used to format scientific publications. Eqn is used to specify the typesetting of mathematical equations. The TeX format is conventionally used with a program called "tex" or "latex" that produces as output a static document. The eqn format is conventionally used with a program called "eqn" that produces as output a static image or description of an equation. (For information on TeX, see Donald E. Knuth, *The TeXbook*, Addison-Wesley, 1986. For information on eqn, see Brian W. Kernighan and Lorinda L. Cherry, "A System for Typesetting Mathematics," *Communications of the ACM* 18:3, March 1975; attached as Exhibit B.)

42. Fourth, Raggett I refers to the invocation of a "shared library or external filter to render the embedded data, e.g. by returning a pixmap". This passage uses several terms of art (in the art of computer science) in ways that teach non-interactivity. "Filter" is a term of art that refers to a type of non-interactive program that translates data from one format to another. "Render" as used by Raggett I is a term of art that refers to the generation of a static image that is to be displayed. "Pixmap" as used by Raggett I is a term of art for a data structure describing an image. "Return" is a term of art that refers to the information produced by a program when that program terminates. A program that has returned something cannot do anything else; for example it cannot provide interactive processing. The use of these four terms of art further teaches the use of static images.

43. Fifth, the only specific example of the use of Raggett's proposed embed tag that is given in Raggett I involves the use of a non-interactive filter which renders static data and then returns. The example depicts the use of the "eqn" program to translate the description of an equation into a static image.

44. Sixth, the discussion of the FIG and ISMAP features in Raggett I is inconsistent with the proposition that Raggett's proposed embed tag

allowed interaction with an embedded object. In Raggett I, an instance of Raggett's proposed embed tag can be placed within a FIG element:

> Instead of the *src* attribute, you can include an EMBED element immediately following the <fig> tag. This is useful for simple graphs, etc. defined in an external format.

(Raggett I at p. 12, emphasis in original) When the FIG element is used in conjunction with the ISMAP parameter (as described in the "Active areas" section of Raggett I, p. 13), the FIG element's display area becomes an image map: any mouse clicks made by the user within the visual depiction of the embedded data will be interpreted by the browser as pertaining to the image-map feature, and will therefore be intercepted by the browser and sent by the browser to the web server. This section of Raggett I teaches that the browser may intercept mouse clicks within the depiction of the embedded data, thereby contradicting the proposition that the embedded data itself can react to mouse clicks.

45. To my knowledge Raggett's proposed embed tag was never implemented. This is confirmed, for example, by Mr. Raggett's trial testimony:

> Q. Sure. I'm sorry. I think you mentioned on direct exam that Mr. Martin's work and Mr. Ang's work and Dr. Doyle's work weren't part of the HTML Plus specification [i.e., of Raggett I].
>
> A. Their work was not part of the specification.
>
> Q. Okay. Now, you understand that they wrote to you in 1994 to describe their use of the embed tag and in fact suggested that you use their version of the embed tag in your upcoming HTML specification, correct?
>
> A. They wrote to me saying that they'd obviously been looking at the HTML Plus specifications, and they were proposing something similar, and I responded to them that at that time there'd been a discussion in the summer of 1993, and at that time the consensus was that the group felt that there were higher priorities and so recommended that we drop the embed mechanism for that moment.

(Eolas v. Microsoft trial transcript at 1884:9-24; see Krueger declaration, Exhibit A)

46. However, if one of ordinary skill in the art (at the time) were asked to implement the Raggett I feature, he would do so by to starting with the existing code for handling IMG tags, and modifying that code. The existing IMG code was able to paint static images into the body of a page, based on an input file that described the image. This code would be modified to invoke an external program, which would return a static image that would then be pasted into the web page in the same manner as in an IMG tag. Such an implementation would not support interactivity within a web browser window.

47. The sentence about "linking to external editors for creating or revising embedded data" refers to the use of external programs by a Web page's author to edit or revise the external data before it is published on the author's Web server.

48. There is nothing in Raggett I to suggest that the "external editors" would provide any display within a web browser window. The editors that were (and still are) conventionally used to create or revise data all run in their own windows; nothing in Raggett I suggests that they would be modified to run within a browser window, or that a browser would be modified to allow the editors to operate in that way. The reference to "linking" to an "external" program refers to the use of a hyperlink or button that the user can click to launch a separate program, as is done with helper applications. (Having the browser automatically invoke an editor wouldn't make sense anyway, since only the page's author would be in a position to edit a copy of the page that anybody else would see, and it wouldn't make sense to invoke an editor automatically when ordinary users had no reason to want to invoke it.)

49. There is nothing in Raggett I that suggests how to provide an interactive program within a browser window – nothing about how to modify a browser to provide such a feature, and nothing about how to modify an editor to work with such a modified browser. No method for doing these things would have been obvious to one of ordinary skill.

### D. The Raggett II Reference

50. Raggett II is a brief email message, written in response to requests for "equation support," "eqn support," and support for "embedded Postscript" in browsers. Equations, eqn data, and embedded postscript are all formats for specifying static data. The requesters ask for support for two rendering programs, eqn and ghostscript, both of which produce static images as output.

51. Raggett II responds by referring to the same functionality described in Raggett I.

52. Raggett II reiterates the teaching of Raggett I about the embedding of static images into Web pages. Raggett II refers to the use of external programs that "render[] foreign formats, e.g. as functions that take a sequence of bytes and return a pixmap." Here again the term of art "render" is used, referring to the creation of a static image.

53. Additionally, the programs are said to "return a pixmap." "Return" is a term of art that refers to the provision of information by a program when that program completes its execution. Therefore, once one of these programs has "return[ed] a pixmap", the program is no longer running and cannot do anything more. In particular, the program cannot provide any interactive functionality, since the program would have stopped running

before the browser even painted the returned static pixmap onto the screen.

54. Raggett II mentions the possibility of implementing the external program as a DLL or dynamically linked library. A DLL is just another way of packaging an executable software application.

55. Raggett II teaches that the programs could be "driven via pipes and stdin/stdout". This refers to a method by which one program invokes another, in such a way that the invoking program can provide input to the invoked program, and can receive any output produced by the invoked program. In this instance, the browser would invoke the external program, would provide the foreign data to the external program, and would receive the external program's output, as a static image.

**E. The Unpatentability Arguments in the Office Action Are Unpersuasive.**

56. From my knowledge of the field, my own personal experience, and the state of the art in 1994, to the extent that a person of ordinary skill in the art was familiar with the teachings of the art cited in the Office Action, I find that the rejection of claims 1 and 6 as obvious is incorrect.

57. For example, the Office Action concludes, incorrectly, that Raggett I teaches interactive processing within a browser window. As described above, Raggett I teaches the use of static content within a browser window, coupled with the use of external editors that appear in separate windows.

58. The core of the Office Action's argument on this point appears in this passage:

> Although Raggett I describes an example where the browser calls a program for rendering an equation in ASCII character format into a pixmap image of the equation, Raggett I does also recognize that more sophisticated browsers can link to external editors for creating or revising embedded data. These external editors that create or revise the embedded data would work in the same way as the simple example of providing equation support. (See Raggett 1: p. 6) However, the ability to create and revise the embedded data allows the user to interactively process the data within the browser window.

(Office Action at 5:42-6:5)

59. The Office Action is incorrect to say that an editor could work "in the same way" that the external rendering programs of Raggett I work. Raggett I's external rendering programs operate by rendering external data to a static image, such as a pixmap, and then returning. Having produced a static image, and having returned (that is, having completed their work), they could not provide interactivity. A program that worked "in the same

way" could not provide editing functionality, or any other form of interactivity.

60. In any case, the editor programs available at the time were incapable of operating in the manner suggested by the Office Action. (They were, of course, capable of being invoked in a separate window.) Raggett I does not suggest the possibility of modifying any editor program. I am not aware of any such description in the prior art of how such modifications might be done; nor does the Office Action point to such a description.

61. If anything the Raggett I and II references teach away from the combinations recited in claims 1 and 6 of the '906 patent. These references teach the use of static web pages, with which the user interacts by moving from page to page, as opposed to the model of the '906 patent where a page can contain a fully interactive object. The two Raggett references teach the inclusion of static images, in various formats, into web pages, but they do not teach interactive processing within a browser window.

62. Finally, I have been told by the patent attorney for Eolas and the Regents that I should consider as part of my obviousness analysis "secondary considerations" such as copying, long felt but unresolved need, properties of the claimed invention, licenses showing industry acceptance of the invention and skepticism of skilled artisans before the invention.

63. I believe there is exceptionally strong "secondary consideration" evidence demonstrating non-obviousness in the case. This evidence includes the failure of others to duplicate the invention. I know of no evidence that either Mr. Raggett or anyone else tried to implement the purportedly obvious combination. In fact, I understand that the "HTML+" syntax described in Raggett I was never implemented.

64. For these reasons, I conclude that the rejection of claims 1 and 6 as being unpatentable is incorrect. The claims of the '906 patent would not have been obvious in view of the references cited in the Office Action.

I declare that all statements made herein of my knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both under 18 U.S.C. Section 1001, and that such willful false statements may jeopardize the validity of the patent.

Dated: May 7, 2004

[illegible]

Edward W. Felten

## EXHIBIT A

# Edward W. Felten

Dept. of Computer Science
Princeton University
35 Olden Street
Princeton NJ 08540
(609) 258-5906
(609) 258-1771 fax
felten@cs.princeton.edu

## Education

Ph.D. in Computer Science and Engineering, University of Washington, 1993.
Dissertation title: "Protocol Compilation: High-Performance Communication for Parallel Programs." Advisors: Edward D. Lazowska and John Zahorjan.
M.S. in Computer Science and Engineering, University of Washington, 1991.
B.S. in Physics, with Honors, California Institute of Technology, 1985.

## Employment

Professor of Computer Science, Princeton University, 2003-present.
Associate Professor of Computer Science, Princeton University, 1999-2003.
Assistant Professor of Computer Science, Princeton University, 1993-99.
Senior Computing Analyst, Caltech Concurrent Computing Project, California Institute of Technology, 1986-1989.

Director, Secure Internet Programming Laboratory, Dept. of Computer Science, Princeton University, 1996-present.

U.S. Dept. of Justice, Antitrust Division: consulting and testimony in Microsoft antitrust case, 1998-2002.
Robins Kaplan, Miller & Ciresi. Consulting and testimony in patent lawsuit, 1998-2003.
Keker & Van Nest. Consulting in intellectual property / free speech lawsuit, 2002.
Electronic Frontier Foundation. Consulting in intellectual property / free speech lawsuits, 2001-present.
Certus Ltd.: consultant in product design and analysis, 2000-2002.
Cigital Inc.: Technical Advisory Board member, 2000-present.
Cloakware Ltd.: Technical Advisory Board member, 2000-present.
Propel.com: Technical Advisory Board member, 2000-2002.
NetCertainty.com: Technical Advisory Board member, 1999-2002.
FullComm LLC: Scientific Advisory Board member, 1999-2001.
Sun Microsystems: Java Security Advisory Board member, 1997-present.
Finjan Software: Technical Advisory Board member, 1997-2002.

EXHIBIT A

International Creative Technologies: consultant in product design and analysis, 1997-98.
Bell Communications Research: consultant in computer security research, 1996-97.

## Honors and Awards

Scientific America 50 Award, 2003.
Alfred P. Sloan Fellowship, 1997.
Emerson Electric, E. Lawrence Keyes Faculty Advancement Award, Princeton University School of Engineering, 1996.
NSF National Young Investigator award, 1994.
Outstanding Paper award, 1997 Symposium on Operating Systems Principles.
Best Paper award, 1995 ACM SIGMETRICS Conference.
AT&T Ph.D. Fellowship, 1991-93.
Mercury Seven Foundation Fellowship, 1991-93.

## Research Interests

Computer security, especially relating to consumer products. Technology law and policy. Internet software. Operating systems. Interaction of security with programming languages and operating systems. Distributed computing. Parallel computing architecture and software.

## Professional Service

### *Professional Societies and Advisory Groups*

ACM Advisory Committee on Security and Privacy, 2002-2003.
DARPA Information Science and Technology (ISAT) advisory group, 2002-present.
Co-chair, ISAT study committee on "Reconciling Security with Privacy," 2001-2002.
National Academy study committee on Foundations of Computer Science, 2001-present.

### *Program Committees*

USENIX General Conference, 2004.
Workshop on Foundations of Computer Security, 2003.
ACM Workshop on Digital Rights Management, 2001.
ACM Conference on Computer and Communications Security, 2001.
ACM Conference on Electronic Commerce, 2001.
Workshop on Security and Privacy in Digital Rights Management, 2001.
Internet Society Symposium on Network and Distributed System Security, 2001.
IEEE Symposium on Security and Privacy, 2000.
USENIX Technical Conference, 2000.
USENIX Windows Systems Conference, 2000.
Internet Society Symposium on Network and Distributed System Security, 2000.

IEEE Symposium on Security and Privacy, 1998.
ACM Conference on Computer and Communications Security, 1998.
USENIX Security Symposium, 1998.
USENIX Technical Conference, 1998.
Symposium on Operating Systems Design and Implementation, 1996.

### *Corporate Advisory Boards*

Sun Microsystems, Java Security Advisory Council.
Cigital Inc.: Technical Advisory Board.
Cloakware Ltd.: Technical Advisory Board.
Propel.com: Technical Advisory Board.
Finjan Software: Technical Advisory Board.
Netcertainty: Technical Advisory Board.
FullComm LLC: Scientific Advisory Board.

## University and Departmental Service

Faculty Advisory Committee on Policy, 2002-present.
Council of the Princeton University Community, 2002-present (Executive Committee)
Faculty Advisory Committee on Athletics, 1998-2000.
Computer Science Academic Advisor, B.S.E. program, class of 1998 (approx. 25 students)
Faculty-Student Committee on Discipline, 1996-98.
Faculty-Student Committee on Discipline, Subcommittee on Sexual Assault and Harrassment, 1996-98.

## Students Advised

### *Ph.D. Advisees:*

Minwen Ji (Ph.D. 2001). Dissertation: Data Distribution for Dynamic Web Content. Researcher at Compaq Systems Research Center.
Dirk Balfanz (Ph.D. 2000). Dissertation: Access Control for Ad Hoc Collaboration. Researcher at Xerox Palo Alto Research Center.
Dan S. Wallach (Ph.D. 1998). Dissertation: A New Approach to Mobile Code Security. Assistant Professor of Computer Science, Rice University.
Robert A. Shillner (Ph.D. expected 2004). Tentative dissertation title: Improving Distributed File Systems using a Shared Logical Disk. Technical staff member at Google.
Michael Schneider (Ph.D. expected 2003). Dissertation topic: Network Defenses against Denial of Service Attacks.

### *Significant Advisory Role:*

Drew Dean (Ph.D. 1998). Advisor: Andrew Appel. Researcher at SRI International.
Stefanos Damianakis, Ph.D. 1998. Advisor: Kai Li. President, Netrics, Inc.
Pei Cao, Ph.D. 1996. Advisor: Kai Li. Assistant Professor of Computer Sciences, University of Wisconsin. On leave at Cisco Systems.

Lujo Bauer, Ph.D. 2003. Advisor: Andrew Appel. Postdoctoral researcher at Carnegie-Mellon University.

## Publications

### *Books and Book Chapters*

[1] Freedom to Tinker. Edward W. Felten. Publication expected, 2004.

[2] Securing Java: Getting Down to Business with Mobile Code. Gary McGraw and Edward W. Felten. John Wiley and Sons, New York 1999.

[3] Java Security: Web Browsers and Beyond. Drew Dean, Edward W. Felten, Dan S. Wallach, and Dirk Balfanz. In "Internet Besieged: Countering Cyberspace Scofflaws," Dorothy E. Denning and Peter J. Denning, eds. ACM Press, New York, 1997.

[4] Java Security: Hostile Applets, Holes and Antidotes. Gary McGraw and Edward Felten. John Wiley and Sons, New York, 1996.

[5] Dynamic Tree Searching. Steve W. Otto and Edward W. Felten. In "High Performance Computing", Gary W. Sabot, ed., Addison Wesley, 1995.

### *Journal Articles*

[6] Mechanisms for Secure Modular Programming in Java. Software – Practice and Experience, 33:461-480, 2003.

[7] The Digital Millennium Copyright Act and its Legacy: A View from the Trenches. Illinois Journal of Law, Technology and Policy, Fall 2002.

[8] DRM and Fair Use: A Skeptical View. Edward W. Felten. Communications of the ACM. April, 2003.

[9] The Security Architecture Formerly Known as Stack Inspection: A Security Mechanism for Language-based Systems. Dan S. Wallach, Edward W. Felten, and Andrew W. Appel. ACM Transactions on Software Engineering and Methodology, 9:4, October 2000.

[10] Statically Scanning Java Code: Finding Security Vulnerabilities. John Viega, Tom Mutdosch, Gary McGraw, and Edward W. Felten. IEEE Software, 17(5), Sept./Oct. 2000.

[11] Client-Server Computing on the SHRIMP Multicomputer. Stefanos N. Damianakis, Angelos Bilas, Cezary Dubnicki, and Edward W. Felten. IEEE Micro 17(1):8-18, February 1997.

[12] Fast RPC on the SHRIMP Virtual Memory Mapped Network Interface. Angelos Bilas and Edward W. Felten. IEEE Transactions on Parallel and Distributed Computing, February 1997.

[13] Implementation and Performance of Integrated Application-Controlled File Caching, Prefetching and Disk Scheduling. Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li. ACM Transactions on Computer Systems, Nov 1996.

[14] Virtual Memory Mapped Network Interface Designs. Matthias A. Blumrich, Cezary Dubnicki, Edward W. Felten, Kai Li, and Malena Mesarina. IEEE Micro, 15(1):21-28, February 1995.

## *Symposium Articles*

[15] Receiver Anonymity via Incomparable Public Keys. Brent R. Waters and Edward W. Felten. ACM Conference on Computer and Communications Security. November 2003.

[16] Attacking an Obfuscated Cipher by Injecting Faults. Matthias Jacob, Dan Boneh, and Edward W. Felten. ACM Workshop on Digital Rights Management, November 2002.

[17] A General and Flexible Access-Control System for the Web. Lujo Bauer, Michael A. Schneider, and Edward W. Felten. 11th USENIX Security Symposium, August 2002.

[18] Informed Consent in the Mozilla Browser: Implementing Value-Sensitive Design. Batya Friedman, Daniel C. Howe, and Edward W. Felten. Hawaii International Conference on System Sciences, January 2002. (Best Paper award, organizational systems track.)

[19] Reading Between the Lines: Lessons from the SDMI Challenge. Scott A. Craver, John P. McGregor, Min Wu, Bede Liu, Adam Stubblefield, Ben Swartzlander, Dan S. Wallach, Drew Dean, and Edward W. Felten. USENIX Security Symposium, August 2001.

[20] Cookies and Web Browser Design: Toward Realizing Informed Consent Online. Lynette I. Millett, Batya Friedman, and Edward W. Felten. Proc. of CHI 2001 Conference on Human Factors in Computing Systems, April 2001.

[21] Timing Attacks on Web Privacy. Edward W. Felten and Michael A. Schneider. Proc. of 7th ACM Conference on Computer and Communications Security, Nov. 2000.

[22] Archipelago: An Island-Based File System for Highly Available and Scalable Internet Services. USENIX Windows Systems Symposium, August 2000.

[23] Proof-Carrying Authentication. Andrew W. Appel and Edward W. Felten. Proc. of 6th ACM Conference on Computer and Communications Security, Nov. 1999.

[24] An Empirical Study of the SHRIMP System. Matthias A. Blumrich, Richard D. Alpert, Yuqun Chen, Douglas W. Clark, Stefanos, N. Damianakis, Cezary Dubnicki, Edward W. Felten, Liviu Iftode, Margaret Martonosi, Robert A. Shillner, and Kai Li. Proc. of 25th International Symposium on Computer Architecture, June 1998.

[25] Performance Measurements for Multithreaded Programs. Minwen Ji, Edward W. Felten, and Kai Li, Proc. of 1998 SIGMETRICS Conference, June 1998.

[26] Understanding Java Stack Inspection. Dan S. Wallach and Edward W. Felten. Proc. of 1998 IEEE Symposium on Security and Privacy, May 1998.

[27] Extensible Security Architectures for Java. Dan S. Wallach, Dirk Balfanz, Drew Dean, and Edward W. Felten. Proc. of 16th ACM Symposium on Operating Systems Principles, Oct. 1997. Outstanding Paper Award.

[28] Web Spoofing: An Internet Con Game. Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Proc. of 20th National Information Systems Security Conference, Oct. 1997.

[29] Reducing Waiting Costs in User-Level Communication. Stefanos N. Damianakis, Yuqun Chen, and Edward W. Felten. Proc. of 11th Intl. Parallel Processing Symposium, April 1997.

[30] Stream Sockets on SHRIMP. Stefanos N. Damianakis, Cezary Dubnicki, and Edward W. Felten. Proc. of 1st Intl. Workshop on Communication and Architectural Support for Network-Based Parallel Computing, February 1997. (Proceedings available as Lecture Notes in Computer Science #1199.)

[31] Early Experience with Message-Passing on the SHRIMP Multicomputer. Richard D. Alpert, Angelos Bilas, Matthias A. Blumrich, Douglas W. Clark, Stefanos Damianakis, Cezary Dubnicki, Edward W. Felten, Liviu Iftode, and Kai Li. Proc. of 23rd Intl. Symposium on Computer Architecture, 1996.

[32] A Trace-Driven Comparison of Algorithms for Parallel Prefetching and Caching. Tracy Kimbrel, Andrew Tomkins, R. Hugo Patterson, Brian N. Bershad, Pei Cao, Edward W. Felten, Garth A. Gibson, Anna R. Karlin, and Kai Li. Proc. of 1996 Symposium on Operating Systems Design and Implementation.

[33] Java Security: From HotJava to Netscape and Beyond. Drew Dean, Edward W. Felten, and Dan S. Wallach. Proc. of 1996 IEEE Symposium on Security and Privacy.

[34] Integrated Parallel Prefetching and Caching. Tracy Kimbrel, Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li. Proc. of 1996 SIGMETRICS Conference.

[35] Software Support for Virtual Memory-Mapped Communication. Cezary Dubnicki, Liviu Iftode, Edward W. Felten, and Kai Li. Proc. of Intl. Parallel Processing Symposium, April 1996.

[36] Protected, User-Level DMA for the SHRIMP Network Interface. Matthias A. Blumrich, Cezary Dubnicki, Edward W. Felten, and Kai Li. Proc. of 2nd Intl. Symposium on High-Performance Computer Architecture, Feb. 1996

[37] Improving Release-Consistent Shared Virtual Memory using Automatic Update. Liviu Iftode, Cezary Dubnicki, Edward W. Felten, and Kai Li. Proc. of 2nd Intl. Symposium on High-Performance Computer Architecture, Feb. 1996

[38] Synchronization for a Multi-Port Frame Buffer on a Mesh-Connected Multicomputer. Bin Wei, Gordon Stoll, Douglas W. Clark, Edward W. Felten, and Kai Li. Parallel Rendering Symposium, Oct. 1995.

[39] A Study of Integrated Prefetching and Caching Strategies. Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li. Proc. of 1995 ACM SIGMETRICS Conference. Best Paper award.

[40] Evaluating Multi-Port Frame Buffer Designs for a Mesh-Connected Multicomputer. Gordon Stoll, Bin Wei, Douglas W. Clark, Edward W. Felten, Kai Li, and Patrick Hanrahan. Proc. of 22nd Intl. Symposium on Computer Architecture.

[41] Implementation and Performance of Application-Controlled File Caching. Pei Cao, Edward W. Felten, and Kai Li. Proc. of 1st Symposium on Operating Systems Design and Implementation, pages 165-178, November 1994.

[42] Application-Controlled File Caching Policies. Pei Cao, Edward W. Felten, and Kai Li. Proc. of USENIX Summer 1994 Technical Conference, pages 171-182, 1994.

[43] Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer. Matthias A. Blumrich, Kai Li, Richard D. Alpert, Cezary Dubnicki, Edward W. Felten, and Jonathan S. Sandberg. Proc. of Intl. Symposium on Computer Architecture, 1994.

[44] Performance Issues in Non-Blocking Synchronization on Shared-Memory Multiprocessors. Juan Alemany and Edward W. Felten. Proceedings of Symposium on Principles of Distributed Computing, 1992.

[45] Improving the Performance of Message-Passing Applications by Multithreading. Edward W. Felten and Dylan McNamee. Proceedings of Scalable High-Performance Computing Conference (SHPCC), 1992.

[46] A Highly Parallel Chess Program. Edward W. Felten and Steve W. Otto. 1988 Conference on Fifth Generation Computer Systems.

## *Other Publications*

[47] Freedom to Tinker weblog, at http://www.freedom-to-tinker.com. Commentary on technology law and policy. Approximately 4000 readers per day.

[48] Secure, Private Proofs of Location. Brent Waters and Edward W. Felten. Submitted for publication, January 2003.

[49] An Efficient Heuristic for Defense Against Distributed Denial of Service Attacks using Route-Based Distributed Packet Filtering. Michael A. Schneider and Edward W. Felten. Submitted for publication, January 2003.

[50] Written testimony to House Commerce Committee, Subcommittee on Courts, the Internet, and Intellectual Property, oversight hearing on "Piracy of Intellectual Property on Peer to Peer Networks." September 2002.

[51] Written testimony to Senate Judiciary Committee hearings on "Competition, Innovation, and Public Policy in the Digital Age: Is the Marketplace Working to Protect Digital Creativity?" March 2002.

[52] Informed Consent Online: A Conceptual Model and Design Principles. Batya Friedman, Edward W. Felten, and Lynette I. Millett. Technical Report 2000-12-2, Dept. of Computer Science and Engineering, University of Washington, Dec. 2000.

[53] Mechanisms for Secure Modular Programming in Java. Lujo Bauer, Andrew W. Appel, and Edward W. Felten. Technical Report CS-TR-603-99, Department of Computer Science, Princeton University, July 1999.

[54] A Java Filter. Dirk Balfanz and Edward W. Felten. Technical Report 567-97, Dept. of Computer Science, Princeton University, October 1997.

[55] Inside RISKS: Webware Security. Edward W. Felten. Communications of the ACM, 40(4):130, 1997.

[56] Simplifying Distributed File Systems Using a Shared Logical Disk.Robert A. Shillner and Edward W. Felten. Princeton University technical report TR-524-96.

[57] Contention and Queueing in an Experimental Multicomputer: Analytical and Simulation-based Results. Wenjia Fang, Edward W. Felten, and Margaret Martonosi. Princeton University technical report TR-508-96.

[58] Design and Implementation of NX Message Passing Using SHRIMP Virtual Memory Mapped Communication. Richard D. Alpert, Cezary Dubnicki, Edward W. Felten, and Kai Li. Princeton University technical report TR-507-96.

[59] Protocol Compilation: High-Performance Communication for Parallel Programs. Edward W. Felten. Ph.D. dissertation, Dept. of Computer Science and Engineering, University of Washington, August 1993.

[60] Building Counting Networks from Larger Balancers. Edward W. Felten, Anthony LaMarca, and Richard Ladner. Univ. of Washington technical report UW-CSE-93-04-09.

[61] The Case for Application-Specific Communication Protocols. Edward W. Felten. Univ. of Washington technical report TR-92-03-11.

[62] A Centralized Token-Based Algorithm for Distributed Mutual Exclusion. Edward W. Felten and Michael Rabinovich. Univ. of Washington technical report TR-92-02-02.

[63] Issues in the Implementation of a Remote Memory Paging System. Edward W. Felten and John Zahorjan. Univ. of Washington technical report TR-91-03-09.

# 831 PH Ex. 15

Application of: Doyle, et al.
Application Num.: 90/006,831
Filed: October 30, 2003
For: Distributed Hypermedia Method for Automatically Invoking External Application Providing Interaction and Display of Embedded Objects Within a Hypermedia Document
Examiner: Caldwell, A.T.
Art unit: 2157

# DECLARATION OF ROBERT J. DOLAN

I, Robert J. Dolan, declare as follows:

1. I have been retained by Eolas Technologies, Inc. and the Regents of the University of California to serve as an expert in the field of marketing. This Declaration is submitted on behalf of Eolas Technologies, Inc. ("Eolas"), the Regents of the University of California, and the inventors of U.S. Patent No. 5,838,906 ("the '906 patent").

## I. INTRODUCTION

2. My name is Robert J. Dolan and I am Dean at the University of Michigan Business School. I am also the Gilbert and Ruth Whitaker Professor at the Michigan Business School. I assumed these positions in July 2001. Prior to joining the University of Michigan Business School, I was a member of the faculty at Harvard Business School. I joined Harvard in 1980 and held the Edward W. Carter Professorship in the Marketing area. See Ex. A (Curriculum Vita).

3. I have been asked to address the arguments presented in the Office Action mailed August 16, 2004 ("the Office Action") in connection with the reexamination of the '906 patent that the claims of the '906 patent are unpatentable as being "obvious." For the reasons described in this Declaration, I believe that the claims of the '906 patent are supported by substantial secondary considerations of non obviousness, namely the commercial success of the Microsoft products that incorporate the technology claimed by the '906 patent.

4. I was an expert witness for Eolas and the University of California during their patent infringement lawsuit against Microsoft Corporation ("Microsoft") in the Northern District of Illinois. As a result, I spent considerable time reviewing and studying the documents, testimony, and information produced (and generated) by the parties to that litigation. The major types of data sources I relied upon were:

   a. internal planning documents, presentations, and memoranda;

   b. e-mail messages;

   c. public statements made by Microsoft, including press releases, annual reports, speeches, and interviews;

   d. previous deposition and trial testimony of Microsoft executives and expert witnesses; and

[redacted]

e. Microsoft's statements and admissions in the litigation between Eolas and the University of California on one hand, and Microsoft on the other hand.

My analysis here focuses on documents admitted as exhibits to the Microsoft litigation or otherwise part of the public record.

## II. MICROSOFT'S COMMERCIAL USE OF THE '906 INVENTION

5. Based upon the jury's verdict in the Microsoft litigation, it is fair to conclude that Microsoft's Internet Explorer ("IE") browser's support for plug-ins, applets, and Active X functionality incorporates the technology claimed in claims 1 and 6 of the '906 patent. The Jury in the Microsoft litigation found that IE, beginning with version 3.0 to the present, includes all of the limitations recited in claims 1 and 6 of the '906 patent. Ex. B (Jury Verdict Form).

6. In the following, the terms "the '906 functionality of IE" and "the '906 functionality of Active X" refer to only those features of IE and Active X, respectively, that are recited in claims 1 and 6 of the '906 patent. Such features include, but are not limited to, automatically invoking an external application, when an embed text format is parsed, to display an object and enable interactive processing of an object in a display area of a hypermedia document being displayed by a browser.

7. Thus, it is appropriate to examine the commercial success of the '906 patent by examining the importance of the '906 functionality of IE and the '906 functionality of Active X to Microsoft.

## III. THE COMMERCIAL SUCCESS OF THE '906 PATENT IS DEMONSTRATED BY THE JURY'S VERDICT AGAINST MICROSOFT

8. The Jury in the Microsoft litigation made the unanimous finding that Microsoft's IE infringed claim 1 and claim 6 of the '906 patent. The jury then awarded Eolas and the University of California a reasonable royalty of $1.47 per unit. The jury applied that royalty rate to 354,124,000 units, for a total damages award of $520,562,280. Ex. B (Jury Verdict Form).

9. Thus, the jury verdict demonstrates substantial units sold by Microsoft of products that contain the '906 functionality of IE. The jury's damages award places a monetary value on the '906 functionality of IE and establishes its commercial success.

## IV. THE COMMERCIAL SUCCESS OF THE '906 PATENT IS DEMONSTRATED BY THE COMMERCIAL SUCCESS OF MICROSOFT'S INTERNET EXPLORER THAT INCORPORATES THE ABILITY TO EMBED INTERACTIVE OBJECTS WITHIN THE BOUNDARY OF A WEB PAGE AS CLAIMED IN THE '906 PATENT

10. The commercial success of the '906 patent is also demonstrated by my analysis of the importance of the '906 functionality of IE to Microsoft. My analysis, set forth below, concludes that the '906 functionality of IE was critical to Microsoft developing a

[redacted]

successful approach to the Internet, and in simultaneously protecting its operating system (Windows) and other business franchises (e.g., Office and Outlook) from the serious threat that existed due to Netscape's browser and concomitant market share. My analysis revealed that, by all accounts, IE 3.0 was a huge success. Indeed, Microsoft employees, customers, web developers and third-party reviewers applauded IE 3.0. More important, however, more people started using IE 3.0 and Microsoft's browser share began to rise. Later versions of IE built upon that success and the features that drove that success – including the '906 functionality of IE and the '906 functionality of Active X – allowing Microsoft to eventually surpass Netscape and become the market leader.

**(a) METHODOLOGY**

11. My analysis focused on the time period beginning in 1995, as Microsoft began to respond to the emergence of the Internet, through November 1998 when the '906 patent issued. I considered the importance to Microsoft of incorporating the '906 functionality of IE in the Windows operating system (of which IE was an integrated part) through IE 3.0, 4.0, and the imminent introduction of 5.0. Over time Microsoft positioned itself as having the "best browser" with a key attribute being the ability to render the widest range of content, including web pages with dynamic, interactive content based on plug-ins, applets, and the '906 functionality of Active X. This positioning allowed Microsoft to win the browser wars and to protect its operating system business.

12. In conducting my analysis, I used a common marketing analysis technique where the "3 C's" of customers, company, and competition are evaluated. Microsoft was, of course, the "company." I focused on Netscape as the primary "competitor," with Sun as a secondary competitor. I analyzed Microsoft's overall position, its integration of the browser into the Windows operating system, its perceptions of the importance of the Internet, the significance of browser share, and the nature of the threat posed to Microsoft by Netscape and others. I also examined the market reception accorded to early generations of IE, e.g., IE 1.0 and 2.0.

13. The competitive analysis focused largely on Netscape due to its early dominance of the browser market and due to Microsoft's views on the significance of the Netscape threat. I focused on the reasons behind of this early market share lead and its implications for Microsoft.

14. The customer analysis had two components. Obviously, one customer type of note is the end user (a consumer or business user) of the Windows operating system and IE. It was also important, however, to recognize the activities of Web site developers. While not purchasers of the product, Microsoft undertook a marketing effort to this group "evangelizing" the IE browser – and the base of end users working with it – in order to encourage Web site developers to "write to" the IE browser functionality.

**(b) ANALYSIS**

15. In the 1993-1994 time frame, the World Wide Web began to transition from laboratories and college campuses into more general and widespread use. In this first phase, the content available to one accessing the web was a large collection of static pages. In other

[redacted]

words, a page could present a user with an easy path to another page (e.g., through a click on a hyperlink), but a user employed a browser basically as the name implies – to browse around a given collection of pages available at that specific point in time. An end user customer could access content, but did not interact with that content.

16. In late 1994, Netscape released Netscape Navigator for browsing. At that same time, a number of other companies including Microsoft, Sun, Apple, IBM, Spry, and Spyglass were also working on browser software. Ex. C (PX 375 at E026853). In early 1995, Microsoft licensed Spyglass's technology and IE 1.0 was integrated into Windows 95 in the summer of 1995.

17. Netscape, however, had already established a clear leadership position in the browser market. By the end of 1995, Microsoft's competitive position was weak as Netscape had attained a large share of web browsing software usage. In June 1995, Netscape pushed beyond the world of a web limited to static pages via an agreement with Macromedia intended to make pages dynamic and interactive. Netscape Navigator 2.0, announced in September 1995, featured inline media plug-ins to afford a new level of interactivity. IE 2.0 – Microsoft's new version of its browser introduced in November 1995 – was recognized as inferior technology compared to Netscape's competitive offering.

18. Faced with this market situation, Bill Gates announced Microsoft's new vision for the Internet. Ex. D (PX 513). The title of Mr. Gates's document setting out the new vision, "The Internet Tidal Wave," stressed the importance of the Internet. Indeed, Mr. Gates described the Internet as "the most important single development to come along since the IBM PC was introduced in 1981." Id. (PX 513 at MSET 0054376). Considering the recognized importance of the Internet, Microsoft's dramatic lagging in web-related technology and market share was particularly significant to Microsoft.

19. In September 1995, a Microsoft senior executive, Paul Maritz, described the seriousness of the situation for Microsoft:

> THE INTERNET
>
> The challenge is that we are in the midst of the "third" digital computer revolution – the first was the invention of mainframes, the second was the invention of the microprocessor, and the third is the communications revolution. Each of these revolutions had distinct winners and losers. IBM was the winner of the first . . . Microsoft is one of the winners of the second revolution, and IBM, the previous winner, suffered.
>
> The Internet is providing the basic standards for the first global, interactive, low cost network. It is the "microprocessor" of this third revolution. Who will provide the "DOS" for this platform?
>
> So far it is Netscape, not Microsoft. Worse still we are not catching up with them. . . .

[redacted]

Ex. E (PX 351 at MSET 0239338).

20. Microsoft's concern was that, just as IBM suffered at the hands of Microsoft during the second digital revolution, Microsoft stood to suffer at the hands of competitor Netscape during the third digital revolution. It was clear to Microsoft that Netscape's browser threatened the Microsoft Windows operating. The Internet and browsers developed for it posed a serious threat to the revenue and profit stream which Microsoft had enjoyed due to Windows. In fact, Ben Slivka, another senior Microsoft executive, wrote a paper entitled "The Web is the Next Platform." Ex. F (PX 350). In a section entitled "Why is the Web a Threat to Windows?," Mr. Slivka focused on the potential of the web to act as an application delivery platform:

> The Web today is a rapidly maturing *application delivery platform*. . . .
>
> My nightmare scenario is that the Web grows into a rich application platform in an operating system-neutral way, and then a company like Siemens or Matsushita comes out with a $500 "WebMachine" that attaches to a TV.

Id. (PX 350 at MSET 0120900) (second emphasis added).

21. Mr. Slivka was not the only person at Microsoft with this concern. Mr. Gates had described the same operating system-neutral phenomenon as Netscape's strategy "to commoditize the underlying operating system." Ex. D (PX 513 at MSET 0054379). This feared "commoditization" would eliminate the position and profits Microsoft had historically enjoyed with its differentiated Windows system; instead, profits would flow to the new leader. As Mr. Slivka noted, "if we don't quickly become the supplier of choice for Internet technology, the Internet will grow and change under someone else's influence, and we risk losing the standard setting role (with the attendant profit margins) we have come to enjoy with MS-DOS and Windows (and Office)." Ex. F (PX 350 at MSET 0120901).

22. Thus, the importance of a browser share and fending off competition at the operating system level became paramount at Microsoft. Microsoft saw this not simply as a "Battle of Browsers" wherein losing would entail just a lost opportunity to add a new revenue stream. Rather, the stakes were much higher; Microsoft saw the Internet as setting out a new marketplace which put the Windows' revenue and profit stream at risk. Gaining market shares with a browser was critical to controlling the evolution of the Internet and unless done quickly, as advocated by Mr. Slivka, the Windows franchise was at risk. In addition, Windows sales drove sales of other Microsoft products. As a result, an inferior browser and the attendant low market share it garnered put these products at risk as well.

23. The importance of browser market shares was hardly a matter of debate at Microsoft. Paul Maritz wrote that "Without browser share, everything is very hard. So job #1 is browser share." Ex. G (PX 588 at MSET 0054473).

[redacted]

24. Based upon the material I reviewed, Microsoft's basic strategy was to develop a browser with a feature set not only allowing it to "catch up" with Netscape, but enabling it to bypass Netscape in functionality. In my opinion, the '906 functionality of IE was essential to that plan.

25. A key feature set of IE 3.0 related to dynamic, interactive content. Netscape already had some capability in this domain on the market and some Web developers were writing to this capability. Thus, IE 3.0 needed to "catch up" by, for example, supporting plug-ins. To induce customers to switch from the then share-dominant competitor, Microsoft had to offer something more than Netscape. Microsoft saw the '906 functionality of Active X as a key differentiator in IE 3.0 as compared to competitor Netscape.

26. Given IE was an integrated part of the Windows operating system, an investment in further developing IE was an investment in improving the quality of the Windows operating system, thereby driving Windows sales.

27. Thus, the stakes of the battle with Netscape and other competitors were clearly recognized. This was not about some product category related to, but distinct from, operating systems. This was about preventing the achievement of Netscape's vision of making Netscape the operating system of the Internet. Netscape's strategy was clear to Microsoft. See Ex. H (PX 585 at MSET 0051473).

28. Microsoft, however, had achieved little success with the first two generations of IE. IE versions 1.0 (launched in summer 1995) and 2.0 (launched in November 1995) were widely regarded as inferior to Netscape's contemporaneous offerings. See Microsoft's statements in Ex. C (PX 375 at E 026849) (admitting that "when Microsoft was offering inferior technology – IE 1.0 and 2.0 – Netscape's usage share remained high"); Ex. I (PX 593 at MSET 0060194) (describing Internet Explorer 1.0 and 2.0 times as a period when "no one used our product because it wasn't a better product than Netscape['s] product").

29. Prior to the launch of IE 3.0, survey information on customer usage reflected Microsoft's perceived inferiority by customers, viz. 52% were using Netscape Navigator as compared to 8% for Microsoft's IE. Ex. J (PX 4062). Netscape was beating IE in the critical market share battle by more than 6 to 1.

30. Interestingly, prior to the release of IE 3.0 in August 1996, Microsoft and in particular, Mr. Gates, had been talking about the importance of supporting dynamic, interactive content. In March 1996, at a joint Microsoft/AOL press conference, Mr. Gates heralded Active X and its ability to move beyond static pages to allow interaction:

> What we're announcing at the developers' conference is a set of technologies we call Active X, and that's the idea of moving from these very static pages that just sit there to pages that interact with you. . . . And we expect that a high percentage of Web pages will use these active extensions. . . . So these are <u>leadership capabilities</u> that the 3.0 browser will include.

Ex. K (PX 377 at MSET 0039808) (emphasis added).

31. IE 3.0's ability to deal with dynamic pages was paramount. For example, the March 1996 press release announcing the prerelease version was headlined "Microsoft Activates the Internet with Prerelease Version of Microsoft Internet Explorer 3.0." In it, Mr. Maritz called IE 3.0 "a major milestone in progressing from a static to dynamic World Wide Web." Ex. L (PX 379 at MSET 0012510).

32. When IE 3.0 was introduced in August 1996, the '906 functionality of Active X was prominently featured as a selling point. For example, one of the introduction events was in San Francisco. Mr. Gates participated in the event and noted ". . . a low key event where IE 3 and the sites using Active X were the stars of the show. The reaction to all of this was fantastic. This is world class work at its finest." Ex. M (PX 389 at MSET 0125888).

33. In fact, the "message" about active content was a key. For example, the marketing plan for IE 3.0 was to focus both on benefits for end users and benefits for developers. Specifically, Microsoft would show that "Microsoft Internet Explorer 3.0, with Active X, puts you 'a step ahead' on the Internet" because it:

- "Provides users the best browsing experience on the Internet (more content, more features, more offers).
- Provides Webmasters/developers next generation platform for creating cool Internet content/apps."

Ex. N (PX 199 at MSET 0244814); <u>see also</u> <u>id</u>. (at MSET 0244823).

34. Indeed, the message that IE 3.0 delivered a dynamic interactive user experience was also showcased in presentations to the industry. For example, in reporting back to Microsoft about a presentation he made on a panel discussion, Mr. Mehdi noted that he presented Microsoft's "standard pitch for Internet Explorer" as follows:

"Messages:

- Humbleness. I said we knew that despite Microsoft's size, we were a small player in terms of browser share. . . . And before we could even start to paint a picture of the great things we could do for customers, <u>we had to at minimum support everything that could be seen today with Netscape and others: Java, Javascript, Plug-ins, NS html enhancements</u>. In IE 3 we do this.
- Focused on customer needs: Trying to <u>solve the challenge of moving from standalone apps and static web pages to dynamic content</u> and rich communication and collaboration. . . .
- IE 3 a step ahead. I walked thru our progress to date. I said IE 1.0 was a release to get us in the browser business and IE 2.0 represented what I

[redacted]

would call a 'catch up' to Netscape feature set. <u>But that IE 3.0 was really the leap frog product for customers and developers in that it layed [sic] out a powerful architecture for developers and provided a very rich feature set for users that would allow them to move to the next step in rich content,</u> communication and collaboration."

Ex. O (PX 363 at MSET 0247873) (emphasis added).

35. And indeed, IE 3.0 did advance over IE 2.0 in a number of ways. For example, Microsoft's Mr. Silverberg designated four categories of improvement: personalization, communication/collaboration and open/secure, in addition to "Cool Interactive Content." Ex. P (PX 531 at MSET 0119502). To match the functionality of Netscape's product, IE 3.0 added support for Netscape's plug-ins. Ex. C (PX 375 at E 026886). IE 3.0 also introduced Active X as another mechanism for dynamic content.

36. Microsoft viewed the '906 functionality of Active X as giving IE an advantage over Netscape: "Active X controls are more capable than Netscape plug-ins and easier to use because they install virtually automatically." Ex. C (PX 375 at E 026886).

37. Thus, the focus was on creating a self-reinforcing process whereby developers would take advantage of the end user's capability of processing dynamic content and end users would be induced to use IE 3.0 because it gave the widest access to Internet content. This made the '906 functionality of Active X a key feature of IE 3.0 and future IE generations driving adoption among end users. Such was the importance of the '906 functionality of Active X to IE 3.0 that in the Microsoft project leader's interview with <u>The Net</u>, Mr. Mehdi characterized this aspect of IE as "the most compelling":

> The most compelling feature will be that you can see the most exciting set of content based on the culmination of Active X, Java and other technologies. With Navigator, you don't get to see Active X content, so you're only going to see part of what's out there on the net.

Ex. Q (PX 390 at MSET 0138394).

38. All of this shows how important Microsoft viewed the browser's ability to render dynamic, interactive pages as a key driver of Windows' sales to consumers, of which IE was an integrated part.

39. As is typical in the computer hardware/software industry, there were many press reviews comparing the new versions of IE and Netscape Navigator. Microsoft's Mr. Chase's summary of these reviews was that:

> The majority of the most influential publications, however, preferred Internet Explorer 3.0 to Netscape Navigator 3.0. Sometimes Internet Explorer 3.0 won by a slight margin, and sometimes it was a very strong winner, but the consensus pick was

[redacted]

> Internet Explorer 3.0. When we talked to users and members of the industry about Internet Explorer 3.0, we could feel how impressed they were with the technology and with our progress since Internet Explorer 2.0. There was a buzz about Internet Explorer 3.0 and how it represented a huge leap forward for Microsoft as well as our customers and partners."

Ex. C (PX 375 at E 026893-94).

40. By the time IE 4.0 was introduced, Microsoft's browser share had increased to approximately 30%. IE 3.0's ability to process sites involving active content --- the '906 functionality of IE --- was critical to this market share increase. At the time of the IE 3.0 launch, 300 web sites already had content requiring Active X support, including popular sites such as Charles Schwab, Chrysler Corporation, FTD Inc., MTV Online, National Geographic Online, and Wall Street Journal Interactive. Ex. R (PX 388 at MSET 0253345, 49-51).

41. In addition, IE 3.0's ability to support the '906 functionality added to its ability to deliver a wide range of sites to the end user. The power of merging Active X support, Netscape plug-ins and Java support was recognized within Microsoft as the "most compelling feature" of IE 3.0, enabling the user to "see the most exciting set of content." Ex. Q (PX 390 at MSET 0138394). For all these reasons, IE 4.0 was also a great success; it built on the functionality and success of IE 3.0. By the second quarter of 1998, Microsoft – with its IE – had attained a 52 to 45 share point advantage over Netscape. See Ex. J (PX 4062).

### (c) CONCLUSION

42. Before the integration of the '906 functionality into IE, Microsoft was dominated by Netscape in browser functionality and usage by customers. According to Microsoft, the "leadership" and "most compelling" feature of IE 3.0 was the '906 functionality of IE. With the introduction of IE 3.0, more and more people began using Microsoft's browser. Indeed, IE began Microsoft's rapid share acceleration process for browsers. Major trade press reviewers changed their recommendations on browser choice from Netscape to Microsoft. Within 14 months of the IE 3.0 introduction, Microsoft's browser share had tripled. IE later rose to market share leadership with new generations of the product – new generations that continued to include the '906 functionality of IE that had proved so important to Microsoft. Ultimately, Microsoft's success with IE was the key to maintaining its profitable Windows franchise and protecting the profits associated with that franchise.

## V. CONCLUSION

43. For the reasons explained above, I conclude that the non obviousness of the claims of the '906 patent is supported by strong secondary considerations of non obviousness, including the substantial commercial success of the '906 functionality of IE.

[redacted]

44. I declare that all statements made herein of my knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both under 18 U.S.C. § 1001, and that such willful false statements may jeopardize the validity of the patent.

Dated: October 8 ,2004

[illegible]

Robert J. Dolan

Word 20108745.1

[redacted]

EXHIBIT A

[redacted]

# ROBERT J. DOLAN

University of Michigan Business School
701 Tappan
Ann Arbor, MI 48109-1234
734-764-1361 phone
734-763-0671 fax
rjdolan@umich.edu

1921 Cambridge
Ann Arbor, MI 48104

## EMPLOYMENT

**University of Michigan Business School**

| | |
|---|---|
| July 2001- present | Dean<br>Gilbert & Ruth Whitaker Professor of Business Administration |
| July 2001 - present | President, William Davidson Institute |

**Harvard University**
**Graduate School of Business Administration**

| | |
|---|---|
| July 1990 - June 2001 | Edward W. Carter Professor of Business Administration |
| July 1985 - July 1990 | Professor of Business Administration (Marketing Area) |
| July 1980 - June 1985 | Associate Professor of Business Administration (Marketing Area) |

Administrative Positions:

- Marketing Area Chairman, 1986-94
- MBA Program Faculty Chairman, 1996-97

**University of Chicago**
**Graduate School of Business**

| | |
|---|---|
| September 1976 - July 1980 | Assistant Professor of Management Science and Marketing (promoted to Associate Professor, effective fall 1980) |

## DEGREES

| | |
|---|---|
| B.A. (1969) | Boston College<br>Chestnut Hill, MA 02167<br>Mathematics (Magna Cum Laude) |
| M.S. (1976) | Graduate School of Management<br>University of Rochester<br>Rochester, NY 14627<br>Business Administration |
| Ph.D. (1977) | Graduate School of Management<br>University of Rochester<br>Rochester, NY 14627 |

Dissertation: "Priority Pricing Models for Congested Systems"

| | |
|---|---|
| M.A. (1986) | Harvard University (Honorary) |

EXHIBIT
tabbies
A

[redacted]

## PUBLICATIONS

### Books

1. Marketing Management: Text and Cases
McGraw Hill-Irwin, (forthcoming)

2. Power Pricing: How Managing Price Transforms the Bottom Line
Coauthor: Hermann Simon (of Simon, Kucher and Partners, Bonn, Germany)
Free Press, 1996
Foreign language versions: Chinese (2000), Korean (1998), Portuguese (1998), German (1997)

3. Managing the New Product Development Process
Addison-Wesley, 1993

4. Strategic Marketing Management (editor)
Harvard Business School Press, 1992
Translated into Spanish and published as La Essencia Del Marketing Estrategia - Vol. 1 and Plan de acción - Vol. II, Editorial Norma S.A.,
Bogota, Colombia, 1995

5. Marketing Management (Coauthors: John Quelch and Thomas Kosnik)
R.D. Irwin, 1993

6. Marketing Management: Principles, Analysis, and Application
(Coauthors: John Quelch and Benson Shapiro)
R.D. Irwin, 1985

7. Marketing Management: Strategy, Planning and Implementation
(Coauthors: John Quelch and Benson Shapiro)
R.D. Irwin, 1985

8. Marketing Management Readings: From Theory to Practice
(Coauthors: John Quelch and Benson Shapiro)
R.D. Irwin, 1985

### Cases (Harvard Business School Case Services)

1. Milford (A), (A1), (B), (C) (with B. Shapiro)
2. Perkin-Elmer Data Systems Group
3. Sealed Air Corporation
4. Federated Industries

10/4/01

[redacted]

5. Henkel Corporation: International Sealants Brand Sista (A), (B)

6. Henkel Corporation: Umbrella Branding Strategy

7. MSA: The Software Company

8. New York Life Pension Department

9. Strategic Industry Model: Emergent Technologies

10. Bayerische Motoren Werke AG (BMW)

11. Northern Telecom (A), (B)

12. Eastman Kodak Company: Funtime Film

13. The Black & Decker Corporation (A), (B), (C), (D)

14. NIKE in the 1990s (A), (B)

15. Launching the BMW Z3 Roadster (with S. Fournier)

16. L'Oréal of Paris: Bringing "Class to Mass" with Plenitude

17. net.Genesis (with R. Lal)

18. Hilary Winsor: A Career in Marketing

19. Abgenix: Xenomouse

<u>Textual Notes</u> (Harvard Business School Case Services)

1. Basic Quantitative Analysis for Marketing

2. Distribution Policy

3. Pricing Policy

4. Marketing Research

5. Research Methods in Marketing: Survey Research

6. Marketing Planning and Organization (with A.J. Silk)

7. Industry Market Strategy

8. Note on Concept Testing

9. Note on Pre-Test Market Models

10/4/01

[redacted]

10. Note on the Market Information Industry

11. Conjoint Analysis: A Manager's Guide

12. Perceptual Mapping: A Manager's Guide

13. Performance Curves: Costs, Prices and Values (with B.P. Shapiro)

14. Industrial Market Research: Beta Site Management

15. Note on Marketing Strategy

16. Note on Low-Tech Marking Math

17. Going to Market

18. Integrated Marketing Communications

19. Analyzing Consumer Perceptions

20. Analyzing Consumer Preferences

21. Pricing and Market Making on the Internet (with Y. Moon)

22. Pricing: A Value-Based Approach

## Journal Articles

1. "Pricing and Market Making on the Internet" Journal of Interactive Marketing

2. "Pricing Technical Products," The Technology Management Handbook, CRC Press 1999.

3. "Price Customization: the Higher Art of Power Pricing," Marketing Management, 1998

4. "How Do You Know When the Price is Right?", Harvard Business Review, September-October 1995.
Reprinted into German as "Der richtige Preis - ewig das Problem?," Harvard Business Manager, January 1996.

5. "Marketing Turnarounds," European Management Journal, September 1995.

6. "Maximizing the Utility of Customer Product Testing: Beta Test Design and Management," Journal of Product Innovation Management, September 1993. (with John Matthews)

7. "Quantity Discounts: Managerial Issues and Research Opportunities," Marketing Science, Winter 1987. (First Runner-Up for TIMS/ORSA Best Marketing Paper of the Year Award.)

8. "Can We Have Rigor and Relevance In Pricing Research?" (with T. Bonoma, V. Crittenden) in Issues in Pricing Research, T. Devinney ed., Lexington Books, 1987.

[redacted]

9. "Dynamic Pricing Strategy: Incorporating Effects of Consumer Price Expectations," (with P. Yoo and K. Rangan), ZfB, 1987.

10. "Models of New Product Diffusion: Competition Against Existing and Potential Firms," (with A. Jeuland, E. Muller), Diffusion Models, Wind and Mahajan, eds., Ballinger Books, 1986.

11. "The Same Make, Many Models Problems: Managing the Product Line," in A Strategic Approach to Business Marketing, R. Spekman and D. Wilson, eds., American Marketing Association, 1985.

12. "A Stimulation Analysis of Alternative Pricing Strategies in a Dynamic Environment," (with D. Clarke), Journal of Business, January 1984.

13. "An Aspect of New Product Planning: Dynamic Pricing," (with A. Jeuland), TIMS Studies in Management Sciences, Marketing Planning Models, A.A. Zoltners ed., 1982.

14. "Definition and Choice of New Product Pricing Strategies," (with D. Clarke) Third Marketing Measurement Conference Proceedings, May 1981.

15. "Pricing Strategies that Adjust to Inflation," Industrial Marketing Management, 1981.

16. "Experience Curves and Dynamic Demand Models: Implications for Optimal Strategies," (with A. Jeuland) Journal of Marketing, Winter, 1981.

17. "An Assessment of the Contribution of Log-Linear Models to Marketing Research," (with R. Blattberg) Journal of Marketing, Spring, 1981.

18. "Models of Competition: A Review of Theory and Empirical Evidence," Review of Marketing 1981, B. Enis and K. Roering, eds.

19. "The Extent of Suboptimality of Myopic Pricing Rules," in Marketing in the 1980s: Changes and Challenges, R. Bagozzi et al., eds., August 1980.

20. "The Panic of the 1980s; It's Pricing," Sales and Marketing Management, June 19, 1980.

21. "The Costs of Model Overfitting," American Marketing Association 1979 Educators' Conference Proceedings, August 1979.

22. "Incentive Mechanisms for Priority Queuing Problems," Bell Journal of Economics, Autumn 1978.

23. "A Normative Model of Industrial Buyer Response to Quantity Discounts," in Research Frontiers in Marketing: Dialogues and Directions, S.C. Jain, ed., August 1978.

24. "Marketing Segmentation via Alternative Discriminant Procedures," in Marketing: The Challenge and Opportunities, August 1975.

**WORKING PAPERS**

10/4/01

[redacted]

1. "Paradoxes: The Prosperity and People of the U.S. Tobacco Industry"

2. "Strategies for Addiction: Measures and Countermeasures"

## VIDEO TAPES

1. "Teaching By the Case Method," (with T.V. Bonoma), American Marketing Association, 1988.

## RESEARCH/EDITORIAL POSITIONS

- Editor, Field Studies Section, Marketing Science, 1989 - 1994

- Member Editorial Review Board
  - Journal of Marketing, 1978 - 1984, 1990 - 1998
  - Marketing Science, 1982 - 1988

- Member
  - Marketing Science Institute Advisory Council, 1986 - 1989
  - Harvard Business School Press Publications Review Board, 1989 – 1992
  - Journal of Public Policy and Marketing, 2001-

## OTHER

- American Marketing Association Doctoral Consortium
  - Coordinator, 1989
  - Faculty Member, 1984, 1986, 1988, 1990

- American Marketing Association Faculty Consortium
  - Faculty Member, 1990, 1992

- Visiting Professorship
  - IESE, Barcelona, Spain
    January – June, 2001

[redacted]

Attachment B

Expert Witness Work Involving Depositions And/Or Trial Testimony in the Past Four Years

1998

- For State of Minnesota in State of Minnesota et. al. vs. Phillip Morris et. al.
- For Commonwealth of Massachusetts et. al. in Commonwealth of Massachusetts et. al. vs. Phillip Morris et. al.
- For Key Pharmaceutical in Key vs. Mylan

1999

- For Medtronic in Medtronic vs. Guidant

2000

- For Compaq in Compa vs. eMachines

[redacted]

# 831 PH Ex. 16

#17 [illegible]
10-12-04

PATENT
Attorney Docket No.: 006-1-1
Client Reference No: 94-108-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | | |
|---|---|---|
| In re reexamination application of: | Examiner: | Caldwell, A. T. |
| DOYLE et al. | Art Unit: | 2151 |
| Application No.: 90/006,831 | Response | |
| Filed: October 30, 2003 | | |
| For: DISTRIBUTED HYPERMEDIA METHOD FOR AUTOMATICALLY INVOKING EXTERNAL APPLICATION PROVIDING INTERACTION AND DISPLAY OF EMBEDDED OBJECTS WITHIN A HYPERMEDIA DOCUMENT | | |

RECEIVED
OCT 1 2 2004
Technology Center 2100

Commissioner for Patents

Sir:

In response to the Office Action mailed 08/16/2004, please consider the following remarks:

## REMARKS

Claims 1-10 have been reexamined and are now pending in the application. Reexamination and reconsideration of all outstanding rejections and objections is requested.

Claims 1 and 6 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.

### 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), the Rule 132 Declaration by Professor Felten submitted with the response filed May 10, 2004 ("Felten I, signed May 7, 2004"), and a Rule 132 Declaration by Robert J. Dolan, Dean at the University of Michigan Business School ("Dolan"). References to these declarations 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 Argument for Claims 1 and 6**

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 6.

**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).

**PART V.** The secondary consideration of commercial success further supports the conclusion of non-obviousness. The attached Declaration of Robert J. Dolan, Dean at the University of Michigan Business School and Gilbert and Ruth Whitaker professor at Michigan Business School ("Dolan") sets forth facts and evidence to legally and factually establish the secondary consideration of commercial success of the invention claimed in claims 1 and 6 of the '906 patent.

a. There is a nexus between the claimed invention and the commercial success.

b. The evidence of commercial success is commensurate with the scope of the '906 claims.

c. The commercial success is derived from the invention.

[redacted]

**DETAILED ARGUMENT**

## A. The Claimed Invention.

The invention, as recited for example in claims 1 and 6, is for use in a system having at least one client workstation and one network server coupled to a network environment.

The claims recite 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 by the browser to identify and locate an executable application external to the hypermedia document.

When an embed text format is parsed by the browser, the executable application 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 1, 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

[redacted]

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.

[redacted]

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 browsing and querying 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:

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

[redacted]

PH_001_0000785808

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.hpl.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.

[redacted]

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 <u>selected</u> by a user for inclusion in the notebook the system will invoke the appropriate application for display in the notebook. <u>Subsequently selecting</u> 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 6 of the '906 claims, or 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 patentee'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.

[redacted]

## D. TRAVERSE

This rejection is respectfully traversed for the following reasons.

The entire Felten declaration II is incorporated herein as an independent traverse of the rejection of claims 1 and 6. 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].

[redacted]

**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 <u>subsequently</u> selecting the data with a mouse will <u>restart</u> the original application so that the data can be edited or updated. [Toye at page 40, first full paragraph]. The fact that

[redacted]

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 6 of the '906 patent, 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 '906 invention were designed to be run in their own windows, under their own control, and contained menubars 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 and 6.**

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 '906 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 6 of the '906 patent. 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 6 of the '906 patent.

Thus at least two elements of claims 1 and 6 of the '906 patent 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 I 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.

[redacted]

**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." ['906 at col. 1, line 61] "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." ['906 at col. 2, line 23]. 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." ['906 at col. 5, lines 25-38].

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 6 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.

[redacted]

## 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 6, lines 23-26, 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." ['906 at col. 1, line 61] "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." ['906 at col. 2, line 23]. 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." ['906 at col. 5, lines 25-38].

The use of the 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 6 of the '906 patent. 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 6 of the '906 patent. 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 '906 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 claim 6. The term "dynamic object" is not used in the '906 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 claim 6 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. ['906 patent at col. 1, lines 30-45].

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.

[redacted]

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 these 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 6 of the '906 patent.

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].

[redacted]

**PART V. The secondary consideration of commercial success further supports the conclusion of non-obviousness. The attached declaration of Robert J. Dolan, Dean at the University of Michigan Business School and Gilbert and Ruth Whitaker professor at Michigan Business School, ("Dolan") sets forth facts and evidence to legally and factually establish the secondary consideration of commercial success of the invention claimed in claims 1 and 6 of the '906 patent.**

**a. There is a nexus between the claimed invention and the commercial success.**

MPEP §716.03 (I) requires that an applicant who is asserting commercial success to support its contention of nonobviousness bears the burden of proof of establishing a nexus between the claimed invention and evidence of commercial success. The term "nexus" designates a factually and legally sufficient connection between the evidence of commercial success and the claimed invention so that the evidence is of probative value in the determination of nonobviousness.

The products analyzed in paragraph (b) of this part are Microsoft Windows and Internet Explorer (IE). [Dolan at paragraph 7]. The Dolan analysis shows that the increase in market share of both Windows and IE 3.0 and later versions is attributed to the incorporation of the '906 functionality of Active X and IE into IE 3.0 and later versions [Dolan at paragraphs 25-41]. IE and ActiveX were found to infringe claims 1 and 6 by a jury in a Federal District Court trial [Dolan at paragraph 5]. This jury finding provides an independent assessment linking the evidence presented in Dolan, establishing commercial success, to claims 1 and 6 of the '906 patent (nexus).

**b. The evidence of commercial success is commensurate with the scope of the '906 claims.**

MPEP §716.03(a) requires that objective evidence of nonobviousness including commercial success must be commensurate with the scope of the claims. The evidence of commercial success must show that the product which has been sold corresponds to the claimed invention, or that whatever commercial success may have occurred is attributable to the product defined by the claims.

The Dolan declaration shows that the commercial success of IE 3.0 and later versions was due to the '906 functionality of ActiveX and IE. [Dolan 25-41]. Accordingly, the evidence of commercial success is commensurate with the scope of claims 1 and 6 of the '906 patent.

**c. The commercial success is derived from the invention.**

MPEP §716.03(b) requires that the commercial success alleged is derived from the invention claimed, in a marketplace where the consumer is free to choose on the basis of objective principles, and that such success is not the result of heavy promotion or advertising, shift in advertising, consumption by purchasers normally tied to applicant or assignee, or other business events extraneous to the merits of the claimed invention.

As established above, the alleged commercial success of IE 3.0 and later versions is based on the '906 functionality of ActiveX and IE. The low market share of earlier versions of IE compared to Netscape Navigator before the incorporation of the '906 functionality of Active X

[redacted]

and IE infers that consumers were free to choose on the basis of objective principles. [Dolan at paragraphs 28 and 29]. The inclusion of the '906 functionality of ActiveX and IE to IE 3.0 and later versions led to independent assessments of the superiority of IE to Netscape Navigator [Dolan at paragraph 39], which resulted in a dramatic and sustained increase of market share of IE due to the inclusion of the '906 functionality of ActiveX and IE in IE 3.0 and later. [Dolan paragraphs 35-41].

Additionally the Jury in the Microsoft litigation independently established a monetary value of $520,562,280 as the monetary value of the incorporation the '906 functionality of Active X and IE into IE 3.0 and later versions. [Dolan paragraphs 5, 8, and 9].

Accordingly, the alleged commercial success is derived from the addition of the '906 functionality of Active X and IE into IE 3.0 and later versions.

[redacted]

## Dependent Claims

Claims 2 and 7 are rejected over the same combination of references as claim 1. The examiner states that claims 2 and 7 have the same scope and only claim 2 is discussed below.

Claim 2 depends on claim 1 and adds the additional limitation that the executable application is a controllable application. The controllable application is controlled on the client workstation via inter-process communication between the browser and the controllable application.

The examiner reasons that Toye teaches a method wherein "said executable application is a controllable application" and the method further comprises the step of "interactively controlling said controllable application on said client workstation via interprocess communications between said browser and said controllable application."

This rejection is respectfully traversed for the following reasons. First, claim 2 depends on claim 1 and is allowable for the same reasons. Further, the executable application in Toye is launched by the user selecting an image in the NoteMail page. [Felten II, at paragraphs 34-35]. This executable application is launched in a separate window and there is no interprocess communication between the browser and the launched application. [Felten II, at paragraphs 36-37].

Accordingly, there is no teaching in the cited references that would have made claims 2 or 7 obvious.

Claims 3 and 8 are rejected over the same combination of references as claim 1. The examiner states that claims 3 and 8 have the same scope and only claim 3 is discussed below.

Claim 3 depends on claims 1 and 2 and further recites the limitation that communications to interactively control the controllable application continue to be exchanged between the controllable application and the browser after the controllable application has been launched.

The examiner states that Toye teaches that selecting the displayed data within a page will restart the original application so that data can be edited or updated without leaving the notebook environment and that the term editing suggests a continued and interactive process controlled by the browser user. It is reasoned that a skilled artisan would reasonably infer that the Mosaic, Berners-Lee, Raggett I and II, and Toye teach a method wherein "communications to interactively control said controllable application continue to be exchanged between the controllable application (i.e., Toye's "appropriate application") and the browser even after the controllable application has been launched".

This rejection is respectfully traversed for the following reasons. First, claim 3 depends on claims 1 and 2 and is allowable for the same reasons. Further, the executable application in Toye is launched by the user selecting an image in the NoteMail page. [Felten II, at paragraphs 34-35]. This executable application is launched in a separate window and there is no interprocess communication between the browser and the launched application. [Felten II, at paragraphs 36-37].

Accordingly, there is no teaching in the cited references that would have made claims 3 or 8 obvious.

[redacted]

Claims 4-5 and 9-10 are rejected under 35 U.S.C. §103(a) as being unpatentable over the same combination as claim 1. The examiner states that claims 4-5 and 9-10 have the same scope and only claims 4-5 are discussed below.

Claim 4 depends on claims 1, 2, and 3 and adds the additional limitations that instructions for controlling the controllable application reside on the network server and that commands can be issued from the client workstation that cause instructions to execute on the network server. Information is sent from the network server to the client workstation in response to the instructions executed on the client workstation and the information is processed on the client workstation to interactively control the controllable application.

The examiner reasons that the combination also described a method wherein additional instructions for controlling said controllable application reside on a network, server citing Toye's teaching that Toye's appropriate application, if not locally resident, will run remotely over the network.

This rejection is respectfully traversed for the following reason. First, claim 4 depends on claims 1-3 and is therefore allowable for the same reasons. Further, the executable application in Toye is launched by the user selecting an image in the NoteMail page. [Felten II, at paragraphs 34-35]. This executable application is launched in a separate window and there is no interprocess communication between the browser and the launched application. [Felten II, at paragraphs 36-37].

Claim 5 depends on claims 1, 2, 3, and 4 and recites the additional limitations that additional instructions for controlling the controllable application reside on the client workstation. This claim is allowable for the same reasons described above.

Accordingly, there is no teaching in the cited references that would have made claims 4-5 or 9-10 obvious.

**Non-applied References**

The cited references that have not been applied against the claims have been reviewed. These references are less relevant to the claims than the applied references and all pending claims are deemed allowable thereover.

## 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

# 831 PH Ex. 17

# '906 Patent Reexamination

Interview with Examiner St. John Courtenay III
August 18, 2005

Michael D. Doyle, Ph.D.
Charles E. Krueger, Attorney

Note: Since a complete traverse of the most recent rejection is already on file (10/12/2004), this presentation is intended only to summarize and emphasize certain portions of that traverse. This presentation is not intended as a substitute for those arguments already submitted.

## 906 Patent Reexamination

- A decade ago, before ease of interactivity had become a key ingredient to the popular success of the Internet, the World Wide Web was in transition from laboratory to dormitory. Far from today's easy-to-use browser technology with seemingly ubiquitous interactivity, the World Wide Web then consisted of a large collection of static text pages through which a user could navigate using a Web browser. As the technology progressed, still images were added to the Web collection; however the user was still only able to access the information, not interact with it. While early Web participants struggled to implement helper applications, researchers at the University of California were already examining the potential of the Web to become a **platform for fully interactive embedded applications**: The '906 invention was born.

## '906 Patent Reexamination

- Claims 1 and 6
  - Scope of the claim
    - Executable application is automatically invoked, when an embed text format is parsed by the browser, in order to display the object and allow in-place interaction while the web page is being displayed
    - Animation of claim 6

## Scope of the Claims

- The scope of patent terms used herein are as utilized during the District Court trial and as affirmed by the Court of Appeals for the Federal Circuit in its 2005 decision.

## The References

- Berners-Lee
  - Provides a specification for the HTML document language
- Raggett I and II
  - Proposed use of a tag called EMBED for specification of static inline images
- Mosaic
  - Early web browser that supported helper applications
- Toye
  - Collaborative engineering notebook system

## The Grounds of Rejection

- States that "The combination of patentee'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"
- States 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."
- States that the patentees' invention would be rendered obvious by "...combining the teachings of the admitted prior art in view of Berners-Lee, Raggett I and Raggett II, by further modifying the combination's static embedded object to be a dynamic embedded object as taught by Toye."

## Summary of Patentee Arguments

- The references do not disclose or teach the features recited in claims 1 and 6.
- The result of a 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
- The obviousness rejection is based on a false premise and therefore reaches a false conclusion
- There is no motivation or teaching in the cited references to combine the references to make the claimed invention obvious
- The references teach away from the proposed combination
- The secondary consideration of commercial success further supports the conclusion of non-obviousness

## Mosaic

- The browser application is utilized as a viewer to read HTML documents published on the World Wide Web.
- The browser retrieves a published Web Page, stores a local copy of the retrieved HTML page source file in a temporary cache, and parses that local copy to form a rendered image of the page which is displayed by the browser to the user.
- The browser allows an author to use the IMG and FIG tags to embed, in a source HTML document, in-line graphic images which are treated as characters when the page is rendered, and which include a src attribute that identifies an image data file external to the document that is retrieved by the browser and rendered into a static graphic image.
- The user could invoke a helper application, which operated in a separate window as an independent program from Mosaic to view data in non-native format. When the helper application became active Mosaic would become inactive.

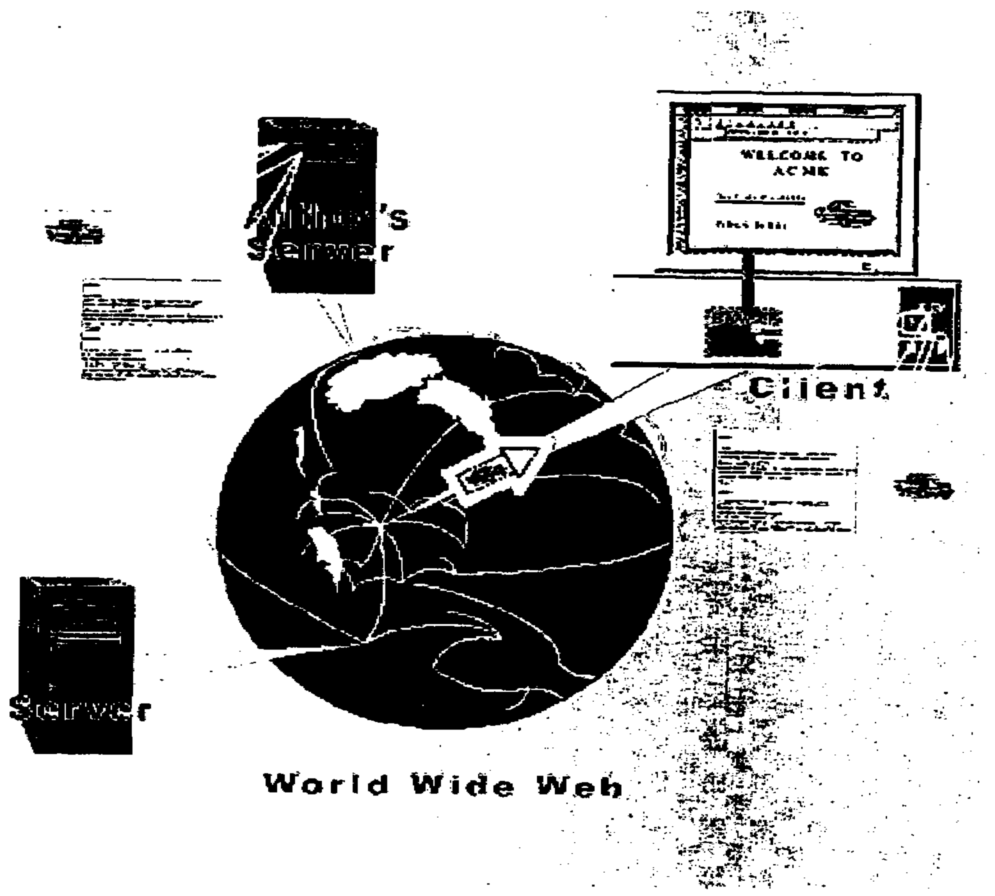
## Berners-Lee

- A specification for the HTML **mark-up language** used by Web authors to describe the structure and desired content of their pages.
- Describes a model in which Web pages are **written by an author**, then distributed by a Web server to a browser, and then **viewed non-interactively as static items** by the browser's user
- The user views a page and then clicks a hyperlink or button, or enters some text in an address field, to view another page.

## Berners-Lee

- "The Berners-Lee reference 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 **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, para. 12, emphasis added]

## Editor vs. Client



## Raggett I

- Is a Web-posted Document entitled HTML+ that proposes a set of slight modifications to Berners-Lee.
- Defines an EMBED tag that extends the concept of the non-interactive display of inlined static images to support foreign data formats that cannot be rendered by the browser itself.
- States that the EMBED tag can be used as a substitute for the src attribute within a FIG tag
- Teaches a **non-interactive** external rendering application that renders embedded data by **returning** a **static** image, such as a pixmap, as it ceases execution.
- Teaches that sophisticated browsers can link to an external editing application that pops up in a window separate from the browser to allow editing of data.

## Raggett II

- Is an email message stating that the EMBED tag of Raggett I has the capability to embed foreign formats, such as equations and encapsulated Postscript, inline in the HTML+ page.
- States that X resources or a config file can be used to bind MIME content type to the rendering application for the format.
- Teaches that the rendering application should operate as "functions that take a sequence of bytes and **return** a pixmap." (emphasis added)
- States that the source file holding the foreign data can be external to the HTML+ source and referenced by a URL.

## HTML+ Specification Inline graphics

- "treated like characters"
  - Images defined by <IMG>, <FIG>, or <EMBED> are all inline graphics
  - All three tags produce **static** pixmaps, that are displayed without the ability to be interactively processed
  - "Sophistocated HTML+ editors should allow authors to modify images using an external editor. Larger images should be specified with the FIG tag"
  - Raggett I teaches here that only the web page **author** would need to modify an inline graphic image.
    - As Berners-Lee teaches, it is the web page **author** who creates and publishes the page content for retrieval by the end-user. Only the **author** can change the source data.

## HTML+ Specification EMBED tag -- Filter

- Raggett I and II's filter application renders data and then **returns** a pixmap
- Execution of the filter ends **before** the browser uses the returned data to render page
- Raggett I gives two examples which result in the **non-interactive** display of static images in the web page
- Filters are **non-interactive**
- Raggett I and II teach implementing the rendering filter applications through UNIX pipes

## HTML+ Specification EMBED tag -- Pipes

- UNIX pipes are treated as files by the calling program
    - As the rendering program ends its execution, it passes the finished image pixmap back to the browser
- In this context, reading the data stream from a pipe is just like reading from a file stream
    - The src attribute specifies a **static** graphic file
    - The ability to substitute an EMBED tag for the src attribute in the FIG tag shows that, to the FIG tag code, EMBED would have behaved like a **static** graphic file
- Since the filter application that renders the static pixmap **must** cease execution prior to painting that image on the screen, that application **cannot, under any circumstances**, be used to interactively control the display of the data while it is being displayed to the user

# HTML+ Specification FIG tag

- Teaches that you can use the EMBED element in place of the src attribute in order to define the image data
  - You can substitute the EMBED-defined pipe, for the src-defined file stream because, to the FIG tag code, they look the same
  - FIG tag is clearly intended for the display of **static** data that **cannot be interactively processed** by the user

## HTML+ Specification FIG tag

- Image maps are a key feature of FIG
  - They provide pre-defined active areas that can be associated with hypertext links
  - A user's click on one of these active areas would cause the browser to fetch a new web document
  - If the image data in an image map changes, the active areas lose their semantic correspondence: they lose their meaning
  - Since Raggett I teaches that EMBED should work with image maps, it **cannot** refer to any method which would allow interactive processing of image data by the user

# HTML+ Specification FIG tag

- A mouse click can **only** mean **one** thing at a time
- The image map feature of the FIG tag would have obviated any ability to interact with EMBED-based images beyond the simple clicking of an image map
- Any mouse clicks on an EMBED-based FIG-tag image would have been captured by the image map code of the FIG tag. The **EMBED**-based image, itself, **could not** have been interactively processed by the user.
- This means that the use of the proposed EMBED tag, itself, was appropriate only for the **non-interactive** display of image data.
- The FIG tag **absolutely excludes** any possibility that EMBED could **ever** be used for in-place interactive control of displayed image data.

## References Teach Away from Interactive Processing

- “Berners-Lee, Raggett I and Raggett II, alone or in combination, do not teach the claim element of enabling interactive processing of an object. **Indeed, they teach away from the provision of interactive processing within the boundaries of a web page.**” [Felten, para. 20, emphasis added]

## Toye

- A system for collaborative editing of engineering documents within an engineering team, using a single object-oriented database to store the documents needed by an engineering workgroup.
- Does not teach the use of a hypermedia browser
- Is not a distributed hypermedia system
- Does not teach use of a distributed hypermedia document
- Does not show automatic invocation to display an object and allow in-place interaction while a document page is being displayed

## Toye

- "Toye is directed to the creation of a system for collaborative editing ... of engineering documents within an engineering team, **using a single object-oriented database** to store the documents needed by an engineering workgroup." [Felten II, para 21]

# The '906 Patent Defines Key Terms Regarding Hypermedia

- **Hypermedia Browser**
  - Is "a browser application, that parses a first distributed hypermedia document to identify text formats included in said distributed hypermedia document," and that parsing is "for responding to predetermined text formats to initiate processing specified by said text formats ['906 patent, 17:3-6]
- **Distributed Hypermedia Document**
  - "A distributed hypertext or hypermedia document typically has many links within it that specify many different data objects located in computers at different geographic locations connected by a network." ['906 patent, 2:59-62]
  - Distributed hypermedia documents contain "text formats" and are parsed "for responding to predetermined text formats to initiate processing specified by said text formats ['906 patent, 17:3-6]

## Toye: No Hypermedia Browser

- Does not teach the use of a hypermedia browser, as that term is defined in the '906 patent
  - ”Toye teaches **no application that parses** distributed hypermedia documents, and it **does not teach other browser-related elements** of the '906 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...” [Felten II, para. 26, emphasis added]
  - “Toye does use the term 'hypermedia browser' but with **a different meaning**. For example, the 'hypermedia browser' of the '906 claims must parse hyperlinks from within a text document, but Toye does not provide that feature.” [Felten II, para. 27, emphasis added]

## Toye: Not Distributed Hypermedia

- Is not a distributed hypermedia system
  - "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, p. 41, emphasis added]
  - "The environment provided by **Toye is not 'distributed'** in the sense of the '906 claims, since it relies on the centralization of a user's document storage in one place. **Toye teaches away from the use of a distributed hypermedia environment."** [Felten II, para. 25, emphasis added]

## Toye: No Automatic Invocation for Interactive Control

- "Toye teaches that NoteMail interacts with an external program by first displaying a **static snapshot** of the external content. If the user clicks on that static snapshot, the external editor application is **restarted** in a separate window." [Felten II, para. 33, emphasis aded]
- "When a data object or file is **selected** for inclusion in the notebook, the system will automatically invoke the appropriate application for displaying that item in the notebook." [Toye, p 40, emphasis added]
- **"Subsequently selecting** displayed data with a mouse will **restart** the original application, so that the data can be edited or updated without leaving the notebook environment. The functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file." [Toye, p 40, emphasis added]
- "Since the user must take specific action to select the data before editing is enabled, the editor is **not 'automatically invoke[d]** ... in order to display said object and enable interactive processing' as required by the '906 claims." [Felten, para. 34, emphasis added]

+ Fel
Dec

## Toye: No Automatic Invocation for Interactive Control

- Although Toye uses the language "automatically invoked," Toye teaches that this action occurs **only as a consequence of the user's active selection**

- Therefore, **Toye does not teach automatic invocation** of an external application to display an object and enable interactive processing of that object within a display area created within a hypermedia document

## The combination does not show the 906 claimed features

- There is no suggestion or teaching in either Toye, the admitted prior art (Mosaic), Berners-Lee, Ragett I or Raggett II, either individually or in combination, 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

- Further, there is no suggestion or teaching in these references, either individually or in combination, 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

# The combination does not show the 906 claimed features

- Toye teaches that NoteMail interacts with an external program by first displaying a **static** snapshot of the external content. If the user clicks on that static snapshot, the external editor application is restarted in a separate window.
  - even if Toye was automatically invoked, the result would only be the **non-interactive display** of a **static** image that would have to be **actively selected by the user in order to launch an editor**
  - even after active selection, the display of the EMBED-specified image would still be limited to being non-interactive, because Toye would be required to **cease execution before the page is rendered**
  - Even if the combination could provide an object which could be automatically invoked to be displayed within the page, while being interactively controllable within the page, the result of a 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

## No Motivation to Combine

- "Neither Toye nor any other reference suggests a combination of Toye with Berners-Lee, Raggett I and Raggett II." [Felten II, para. 39]
  - Toye's "openness and flexibility" language does **not teach nor suggest** that the references could or should be combined
    - "Toye is simply asserting that its system has advantages over other engineering collaboration systems. Toye offers more than static bitmaps; it offers also the ability to click on those bitmaps and launch an external application (in a separate window, as discussed above). Toye offers more than just FrameMaker-compatible formats. 'Openness and flexibility' are little more than buzzwords here. Nothing in this paragraph would teach a PHOSA that Toye could or should be combined with a web browser." [Felten II, para. 41]

## No Motivation to Combine

- Fundamental problems solved by the web and by Toye teach away from a combination
- It is required to consider the references in their entirety, including those portions that argue against obviousness
- When considered in their entirety, it becomes clear that Toye, on the one hand, and the combination of Mosaic, Berners-Lee, Raggett I and II, on the other hand, represent **fundamentally different and incompatible** paradigms

## The References Teach Away from a Combination

- "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 all data relating to documents in a **centralized** object-oriented database; Berners-Lee teaches that documents and data objects can be retrieved from **anywhere and everywhere** on the Internet"
- "Toye teaches that the document text is **not parsed;** Berners-Lee teaches that **parsing of the document text is central** to the functioning of the Web"
- "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**

[Felten II, para 42, emphasis added]

## The References Teach Away from a Combination

- "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 URL**s, which contain location" information
- **"Importantly, Toye teaches away from the use of distributed hypermedia, which is the central idea of Berners-Lee"**

[Felten II, para 42-43, emphasis added]

## Secondary Considerations

- The **unprecedented commercial success** of Microsoft Windows supports the conclusion of non-obviousness
- There is an **indisputable nexus** between the invention and Windows' commercial success
    - Microsoft Internet Explorer was **found to infringe** the 906 patent
    - The Infringement verdict was upheld by the CAFC
- The evidence of commercial success is commensurate with the scope of the '906 claims
    - The US portion of the $565 million damages judgement was **not appealed** by Microsoft
    - The foreign-sales portion of the award was upheld by the CAFC
- The "commercial success is **derived from** the addition of the **906** functionality of ActiveX and IE" into MS Windows

# Summary of Patentees' Arguments

- The references do not disclose or teach the features recited in claims 1 and 6.
  - Raggett teaches the use of an external filter program that **must finish** executing **before** the image it generates is displayed
  - Raggett's FIG tag **absolutely excludes** any possibility that EMBED could **ever** be used to specify for in-place interactive control of displayed image data
  - Toye teaches that NoteMail interacts with an external program by the **non-interactive** display of a **static** snapshot of the external content. If the user clicks on that static snapshot, the external editor application is **restarted** in a **separate** window.
- Even if the combination could provide the display and in-place interactive control of a data object within the hypermedia document page, the result of a combination of Toye with the combination of Mosaic, Berners-Lee, Raggett I and II would render the latter combination **inoperable for its intended purpose**

# Summary of Patentees' Arguments

- The obviousness rejection is based on a **false premise** and therefore reaches a **false conclusion**
  - Toye is **not** a **distributed hypermedia** system
  - Toye **doesn't teach automatic invocation** to provide display and in-place interactive control of a data object within a hypermedia document page
- There is **no motivation or teaching** in the cited references to combine the references to make the claimed invention obvious
- The references **teach away** from the proposed combination
- The secondary consideration of **commercial success** further supports the conclusion of non-obviousness
  - The "commercial success is **derived from** the addition of **the 906 functionality** of ActiveX and IE" into MS Windows

# 831 PH Ex. 18

09/15/05

PATENT
Attorney Docket No.: 006-1-1
Client Reference No: 94-108-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | |
|---|---|
| In re reexamination application of: | Examiner: St. John Courtenay III. |
| DOYLE et al. | Art Unit: 2194 |
| Application No.: 90/006,831 | Interview Summary |
| Filed: October 30, 2003 | |
| For: DISTRIBUTED HYPERMEDIA METHOD FOR AUTOMATICALLY INVOKING EXTERNAL APPLICATION PROVIDING INTERACTION AND DISPLAY OF EMBEDDED OBJECTS WITHIN A HYPERMEDIA DOCUMENT | |

## OFFICE INTERVIEW OF 18 AUGUST 2005

Attending the interview representing the assignee and exclusive licensee were Dr. Michael D. Doyle, one of the inventors, and Charles E. Krueger, the attorney of record, and representing the Patent Office were Examiners St. John Courtenay III and his Supervisor Mark Reinhardt.

The subject matter discussed related to the rejection of claims 1 and 6 over the Applicants' Admitted Prior Art, Berners-Lee, and Raggett I and II, and Toye. The issues were discussed in connection with a set of slides which are attached hereto. The cited, but not applied, reference Media Mosaic was also discussed.

The examiner stated that the OPLA was considering whether the Viola code, submitted by applicants in an IDS in the reexam proceeding, should be considered as a publication.

Dr. Doyle mentioned that his recollection was that there was trial testimony related to how the Viola code files were posted to an ftp server and then removed from the server after a person was supposed to have downloaded them. He then stated that OPLA should read the testimony itself to confirm what was said at trial.

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

66155 U.S.

09/15/05

PTO/SB/21 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

# TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

| | |
|---|---|
| Application Number | 90/006, 83 |
| Filing Date | 10/30/2003 |
| First Named Inventor | Doyle |
| Art Unit | 2184 |
| Examiner Name | St John Courtenay III |
| Attorney Docket Number | 006-1-1 |
| Total Number of Pages in This Submission | |

## ENCLOSURES (Check all that apply)

| | | |
|---|---|---|
| ☐ Fee Transmittal Form | ☐ Drawing(s) | ☐ After Allowance Communication to TC |
| ☐ Fee Attached | ☐ Licensing-related Papers | ☐ Appeal Communication to Board of Appeals and Interferences |
| ☐ Amendment/Reply | ☐ Petition | ☐ Appeal Communication to TC (Appeal Notice, Brief, Reply Brief) |
| ☐ After Final | ☐ Petition to Convert to a Provisional Application | ☐ Proprietary Information |
| ☐ Affidavits/declaration(s) | ☐ Power of Attorney, Revocation Change of Correspondence Address | ☐ Status Letter |
| ☐ Extension of Time Request | ☐ Terminal Disclaimer | ✓ Other Enclosure(s) (please Identify below): |
| ☐ Express Abandonment Request | ☐ Request for Refund | Return Postcard. |
| ☐ Information Disclosure Statement | ☐ CD, Number of CD(s) ______ | Interview ~~Sum~~ Summary |
| ☐ Certified Copy of Priority Document(s) | ☐ Landscape Table on CD | |
| ☐ Reply to Missing Parts/ Incomplete Application | Remarks | |
| ☐ Reply to Missing Parts under 37 CFR 1.52 or 1.53 | | |

## SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

| | | | |
|---|---|---|---|
| Firm Name | LAW OFFICE OF CHARLES E. KRUEGER | | |
| Signature | [illegible] | | |
| Printed name | Charles E. Krueger | | |
| Date | | Reg. No. | 30,077 |

## CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

| | | | |
|---|---|---|---|
| Signature | [illegible] | | |
| Typed or printed name | Charles ~~Sharon D.~~ Krueger | Date | 9/15/2005 |

# 831 PH Ex. 19

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 90/006,831 | 10/30/2003 | 5838906 | | 9718 |

30080 7590 09/27/2005

LAW OFFICE OF CHARLES E. KRUEGER
P.O. BOX 5607
WALNUT CREEK, CA 94596-1607

| EXAMINER | |
|---|---|
| ST. John Courtenay III | |
| ART UNIT | PAPER NUMBER |
| 3992 | |

DATE MAILED: 09/27/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**DO NOT USE IN PALM PRINTER**

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

# *EX PARTE* REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. 90/006,831.

PATENT NO. 5838906.

ART UNIT 3992.

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

PTOL-465 (Rev.07-04)

| Notice of Intent to Issue Ex Parte Reexamination Certificate | Control No. | Patent Under Reexamination | |
|---|---|---|---|
| | 90/006,831 | 5838906 | |
| | Examiner | Art Unit | |
| | St. John Courtenay III | 3992 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

1. ☒ Prosecution on the merits is (or remains) closed in this *ex parte* reexamination proceeding. This proceeding is subject to reopening at the initiative of the Office or upon petition. *Cf.* 37 CFR 1.313(a). A Certificate will be issued in view of
   (a) ☒ Patent owner's communication(s) filed: *12 October 2004*.
   (b) ☐ Patent owner's late response filed: _____.
   (c) ☐ Patent owner's failure to file an appropriate response to the Office action mailed: _____.
   (d) ☐ Patent owner's failure to timely file an Appeal Brief (37 CFR 41.31).
   (e) ☐ Other: _____.

   Status of *Ex Parte* Reexamination:
   (f) Change in the Specification: ☐ Yes ☐ No
   (g) Change in the Drawing(s): ☐ Yes ☐ No
   (h) Status of the Claim(s):
      (1) Patent claim(s) confirmed: *1-10*.
      (2) Patent claim(s) amended (including dependent on amended claim(s)): _____
      (3) Patent claim(s) cancelled: _____.
      (4) Newly presented claim(s) patentable: _____.
      (5) Newly presented cancelled claims: _____.

2. ☒ Note the attached statement of reasons for patentability and/or confirmation. Any comments considered necessary by patent owner regarding reasons for patentability and/or confirmation must be submitted promptly to avoid processing delays. Such submission(s) should be labeled: "Comments On Statement of Reasons for Patentability and/or Confirmation."

3. ☒ Note attached NOTICE OF REFERENCES CITED (PTO-892).

4. ☐ Note attached LIST OF REFERENCES CITED (PTO-1449 or PTO/SB/08).

5. ☐ The drawing correction request filed on _____ is: ☐ approved ☐ disapproved.

6. ☐ Acknowledgment is made of the priority claim under 35 U.S.C. § 119(a)-(d) or (f).
   a)☐ All b)☐ Some* c)☐ None of the certified copies have
   ☐ been received.
   ☐ not been received.
   ☐ been filed in Application No. _____.
   ☐ been filed in reexamination Control No. _____.
   ☐ been received by the International Bureau in PCT Application No. _____.

   * Certified copies not received: _____.

7. ☐ Note attached Examiner's Amendment.

8. ☒ Note attached Interview Summary (PTO-474).

9. ☐ Other: _____.

[illegible] III

ST. JOHN COURTENAY III
PRIMARY EXAMINER

St. John Courtenay III
Primary Examiner
Art Unit: 3992

cc: Requester (if third party requester)

|  |  |  |
|---|---|---|
| **Ex Parte Reexamination Interview Summary** | **Control No.** 90/006,831 | **Patent Under Reexamination** 5838906 |
|  | **Examiner** St. John Courtenay III | **Art Unit** 3992 |

All participants (USPTO personnel, patent owner, patent owner's representative):

(1) *St. John Courtenay III* (3) *Michael D. Doyle*

(2) *Mark Reinhart* (4) *Charles Krueger*

Date of Interview: *18 August 2005*

Type: a)☐ Telephonic b)☐ Video Conference
c)☒ Personal (copy given to: 1)☐ patent owner 2)☒ patent owner's representative)

Exhibit shown or demonstration conducted: d)☒ Yes e)☐ No.
If Yes, brief description: *Powerpoint presentation of Patent Owner's arguments.*

Agreement with respect to the claims f)☐ was reached. g)☐ was not reached. h)☒ N/A.
Any other agreement(s) are set forth below under "Description of the general nature of what was agreed to..."

Claim(s) discussed: *1 and 6*.

Identification of prior art discussed: *Mosaic (APA), Berners-Lee, Raggett I & II, and Toye*.

Description of the general nature of what was agreed to if an agreement was reached, or any other comments:
*The Patent Owner presented a Powerpoint presentation summarizing the Patent Owner's arguments of record. The Examiner informed the patent owner that OPLA was reviewing the Viola code to determine if it should be considered as a prior art publication.*

(A fuller description, if necessary, and a copy of the amendments which the examiner agreed would render the claims patentable, if available, must be attached. Also, where no copy of the amendments that would render the claims patentable is available, a summary thereof must be attached.)

A FORMAL WRITTEN RESPONSE TO THE LAST OFFICE ACTION MUST INCLUDE PATENT OWNER'S STATEMENT OF THE SUBSTANCE OF THE INTERVIEW. (See MPEP § 2281). IF A RESPONSE TO THE LAST OFFICE ACTION HAS ALREADY BEEN FILED, THEN PATENT OWNER IS GIVEN **ONE MONTH** FROM THIS INTERVIEW DATE TO PROVIDE THE MANDATORY STATEMENT OF THE SUBSTANCE OF THE INTERVIEW (37 CFR 1.560(b)). THE REQUIREMENT FOR PATENT OWNER'S STATEMENT CAN NOT BE WAIVED. **EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.550(c).**

ST. JOHN COURTENAY III
PRIMARY EXAMINER

[illegible]

Examiner's signature, if required

cc: Requester (if third party requester)

UNITED STATES DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE

# REEXAMINATION

## REASONS FOR PATENTABILITY / CONFIRMATION

Reexamination Control No. 90/006,831 Attachment to Paper No. 20050823.

Art Unit 3992.

See attached "Examiner's Statement of Reasons for Patentability / Confirmation."

ST. JOHN COURTENAY III
PRIMARY EXAMINER

[illegible]
(Examiner's Signature)

PTOL-476 (Rev. 03-98)

## Examiner's Statement of Reasons for Patentability and/or Confirmation

The following is an Examiner's statement of reasons for patentability and/or confirmation of the claims found patentable in this reexamination proceeding.

### Summary

At the outset, it is noted that the previous Examiner of record admitted in making the rejection under 35 U.S.C. §103 of independent claims 1 and 6 that the cited four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II, does not explicitly teach a method that enables interactive processing of an object:

> The combination of patentee'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, with distributed hypermedia documents [see Office Action mailed Oct. 16, 2004, page 6, lines 18-21].

The previous Examiner then applied a fifth reference (Toye) to the combination and asserted:

> 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 (See Toye: p. 40 description of NoteMail, particularly p. 40, col. 2, first paragraph).

An Examiner's statement of reasons for confirmation and/or patentability is set forth below in the form of a reply to the Patent Owner's detailed arguments of record. The Patent Owner's arguments are shown in *italics* below. In addition, the "DX37" Viola code has been considered by the PTO as a prior art publication. The Viola code issue is addressed at the end of the response to the Patent Owner's detailed argument.

## Examiner's Response to Patent Owner's Detailed Argument

*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.*

In response, the Examiner finds the Patent Owner's argument I(a) persuasive for at least the following reasons:

As acknowledged by the previous Examiner, the cited four-way combination of the patent owner's admitted prior art (APA), 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, with distributed hypermedia documents" [see Office Action mailed Oct. 16, 2004, page 6, lines 18-21].

During patent examination, the pending claims must be "given their broadest reasonable interpretation consistent with the specification." In re Hyatt, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000).

Accordingly, with respect to the scope of the claimed "interactive processing," the Examiner must apply the broadest reasonable interpretation consistent with the specification.

To be consistent with the specification, the claimed "interactive processing" necessarily requires some capability of ongoing real-time manipulation and control by the user of the object displayed within the browser-controlled window.

In particular, the claimed "interactive processing," when properly construed in a manner consistent with the specification, requires:

> "**Interprocess communication** between the **hypermedia browser** and the **embedded application program** is **ongoing** after the program object has been launched" [see instant '906 patent, col. 7, lines 1-4].

Static objects disclosed by the prior art of record, such as graphical images of mathematical formulas (see e.g., the use of the EMBED tag in Raggett I at the bottom of page 6) are incapable of providing "interactive processing" as required by the instant '906 claims because the application that renders the static object terminates after the rendering step and prior to the complete display of the web page.

With respect to prior art of record that uses colored or otherwise identifiable active areas superimposed on a coordinate grid of a static image map (e.g., see the use of the "ismap" attribute and "<figt " tag in Raggett I - see "Active areas" on page 13; see also U.S. Patent 4,847,604 to Doyle), these "map" images are created by an executable rendering application that

generates the static "map" image and then terminates prior to the complete display of the web page.

The aforementioned prior art "map" images are static in the sense that the user cannot interactively change the appearance of the "map" image, but are also active in the sense that the user can interactively click on an active region or area within the map and trigger a URL that is invoked by the web browser application.

Significantly, with respect to active maps and the like, it is the browser application (i.e., not an executable application separate from the browser application) that makes the active areas "interactive" by waiting for user input, typically in the form of a mouse click [see e.g., Raggett I, page 13, 1st sentence under "Active areas"].

Because the aforementioned prior art executable rendering applications terminate after generating the static image, it is axiomatic that there is no ongoing interprocess communication between the browser and the executable application. Therefore, there is no ongoing real-time manipulation and control by the user of the object displayed within the browser-controlled window, as required by the instant '906 claims when the claim element "interactive processing" is properly construed in a manner consistent with the specification of the '906 patent.

The instant '906 patent specification makes liberal use of the term "interactive" as being synonymous with "manipulate" and "control" in an ongoing real-time setting:

See '906 Patent, col. 6, lines 40-47:

> Thus, it is desirable to have a system that allows a user at a small client computer connected to the Internet to locate, retrieve and **manipulate** data objects when the data objects are bandwidth-intensive and compute-intensive. Further, it is desirable to allow a user to **manipulate** data objects in an **interactive** way to provide the user with a better understanding of information presented and to allow the user to accomplish a wider variety of tasks.

See '906 Patent, col. 6, lines 50-62:

> The present invention provides a method for running embedded program objects in a computer network environment. The method includes the steps of providing at least one client workstation and one network server coupled to the network environment where the network environment is a distributed hypermedia environment; displaying, on the client workstation, a portion of a hypermedia document received over the network from the server, where the hypermedia document includes an **embedded controllable application**; and **interactively controlling** the **embedded controllable application** from the client workstation via communication sent over the distributed hypermedia environment.

See '906 Patent, col. 6, lines 63-67 cont'd col. 7, lines 1-6:

> The present invention allows a user at a client computer connected to a network to locate, retrieve and **manipulate objects** in an **interactive way**. The invention not only allows the user to use a hypermedia format to locate and retrieve program objects, but also allows the user to **interact** with an application program located at a remote computer. **Interprocess communication between the hypermedia browser and the embedded application program is ONGOING after the program object has been launched**. The user is able to use a vast amount of computing power beyond that which is contained in the user's client computer.

See '906 Patent, col. 9, line 66 cont'd col. 10, lines 1-16:

> After application client 210 receives the multidimensional data object 216, application client 210 executes instructions to display the multidimensional embryo data on the display screen to a user of the client computer 200. The user is then able to **interactively operate controls to recompute different views for the image data.** In a preferred embodiment, a control window is displayed within, or adjacent to, a window generated by browser client 208 that contains a display of hypermedia document 212. An example of such display is discussed below in connection with FIG. 9. Thus, **the user is able to interactively manipulate a multidimensional image object** by means of the present invention. In order to make application client 210 integral with displays created by browser client 208, both the browser client and

> the application client must be in communication with each other, as shown by the arrow connecting the two within client computer 200. The manner of communication is through an application program interface (API), discussed below.

See '906 Patent, col. 10, lines 47-56:

> In the present example where a multidimensional image object representing medical data for an embryo is being viewed, application server 220 could perform much of the viewing transformation and volume rendering calculations to **allow a user to interactively view** the embryo data at their client computer display screen. In a preferred embodiment, application client 210 **receives signals from a user input device** at the user's client computer 200. **An example of such input would be to rotate the embryo image from a current position to a new position from the user's point of view**.

See '906 Patent, col. 16, lines 18-20.

> FIG. 9 is a screen display of the invention showing an **interactive application object (in this case a three dimensional image object)** in a window within a browser window. In FIG. 9, the browser is NCSA Mosaic version 2.4. The processes VIS, Panel and VRServer work as discussed above. FIG. 9 shows screen display 356 Mosaic window 350 containing image window 352 and a portion of a panel window 354. Note that <u>image window 352 is within Mosaic window 350</u> while panel window 354 is external to Mosaic window 350. Another possibility is to have panel window 354 within Mosaic window 350. By **using the controls** in panel window 354 **the user is able to manipulate the image within image window 352 in REAL TIME to perform such operations as scaling, rotation, translation, color map selection, etc.**

The Examiner submits that this interpretation is reasonable and also consistent with the interpretation that those skilled in the art would reach. See <u>In re Cortright</u>, 165 F.3d 1353, 1359, 49 USPQ2d 1464, 1468 (Fed. Cir. 1999), "The broadest reasonable interpretation of the claims must also be consistent with the interpretation that those skilled in the art would reach."

The above discussion does not mean that the use of static objects precludes interactivity. One may reasonably argue that the use of static graphical images that contain superimposed active areas or sections (e.g., through the use of the "ismap" attribute and "<figt " tag in Raggett I, *supra*) enable "interactive processing" in the sense that when a user clicks the mouse over an active area, a URL call to a server is generated by the browser; however, this is not the same kind of "interactive processing" required by instant claims 1 and 6 of the '906 patent.

In the case of the Raggett I "*ismap*" attribute, Raggett explicitly discloses:

> "The *ismap* attribute causes the browser to send mouse clicks on the figure, back to the server using the selected coordinate scheme" [see Raggett I, page 13, 1st sentence under "Active areas"].

As is clearly indicated by Raggett I, it is the browser application that responds to the mouse click that occurs over an active region identified by a coordinate scheme superimposed over a static graphical image. Thus, in the case of Raggett I and active map areas in general (e.g., using the "ismap" attribute and "<figt " tag), it is the browser application that provides the interactivity.

In contrast, the instant '906 claims explicitly require the "interactive processing" to be enabled by an "executable application" that is a separate application from the browser application.

The instant claimed '906 "executable application" that provides the claimed "interactive processing" is invoked not in response to a user event detected by the browser (as in the case of Raggett I, *supra)*, but rather in response to the browser application parsing an "embed text format" (i.e., an "EMBED" tag, see col. 12, line 60, '906 patent) that is detected within the hypermedia document when the hypermedia document is first loaded by the browser.

Significantly, the instant claimed "interactive processing" of the '906 patent begins at the moment the browser application parses an "embed text format" detected within the hypermedia document. The web browser invokes the claimed "executable application" immediately after an "EMBED" tag is parsed and before the hypermedia document is completely displayed in the browser-controlled window. The invoked "executable application" enables the claimed "interactive processing."

Instant '906 independent claims 1 and 6 therefore require an <u>operative coupling</u> between the claimed "executable application" and the claimed "interactive processing" such that the claimed "interactive processing" must be enabled by an "executable application" that meets five explicitly claimed requirements:

1. The executable application must be external to the first distributed multimedia document.
2. The executable application must be automatically invoked by the browser application when the "embed text format" is <u>parsed</u> by the browser application.
3. The executable application must execute on the client workstation.
4. The executable application must display the object within the display area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser-controlled window.

5. The executable application must enable interactive processing of the object within the display area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser-controlled window.

Because the admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II four-way combination displays or renders a static image and then terminates, "interactive processing" as used in the instant claims is precluded by the four-way combination.

As discussed *supra,* a proper construction of the claimed "interactive processing" necessarily requires some capability of ongoing real-time manipulation and control by the user that is applied to the object displayed within the first browser-controlled window. It is axiomatic that an executable application that terminates is incapable of providing the type of "interactive processing" required by instant '906 independent claims 1 and 6.

In particular, executable application requirement #5, *supra,* is clearly not met by the cited four-way combination of admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II, with respect to the operative coupling required between the claimed "executable application" and claimed "interactive processing."

## THE TOYE REFERENCE

The Examiner finds the Patent Owner's argument (as supported by the Felten II affidavit, §§33-35) persuasive that Toye teaches the use of an image or icon that represents a file or data object displayed within a "NoteMail" page, and that the image or icon consists of a "static snapshot" of the external content. Interactive processing is enabled only after a user manually clicks on the "static snapshot" image to launch an external editor program.

Toye discloses manual selection by the user to enable interactivity:

> "Subsequently selecting the displayed data with a mouse will **restart** the original application, so that the data can be edited or updated without leaving the notebook environment" [See Toye, p. 40, 2nd column, 2nd paragraph].

Significantly, Toye discloses functionality similar to a file manager:

> "The functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file" [p. 40, 2nd column, 2nd paragraph].

The Examiner concurs with the Patent Owner's contention that no ongoing interaction with the data can occur unless the "appropriate application" is manually started or restarted by the user to enable interaction with the data displayed as a static "snapshot image" or icon within a "NoteMail" page.

The Examiner concurs that automatic invoking, as taught by Toye, is the result of manual user selection with a mouse of a "static snapshot" image that automatically launches the "appropriate application" to edit the data object. This approach appears to be similar to the method employed by conventional file manager programs that implement file type association to invoke the appropriate application when the user clicks on the filename or file icon.

Accordingly, the Examiner concurs with the Patent Owner that Toye teaches away from automatic invocation of an external application when a document is <u>parsed</u> to enable interactive processing of the object, and instead teaches that an object must be <u>selected by a mouse</u> to invoke an application to enable interactive processing.

> *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 response, the Examiner finds the Patent Owner's argument I(b) persuasive for at least the following reasons:

The Examiner concurs with the Patent Owner's argument regarding the Raggett I & II EMBED tag that is located at a first location in a hypermedia document. When the EMBED tag is parsed, a rendering application is invoked that returns a STATIC graphical image to be displayed within the browser window at the first location, and then the rendering application <u>terminates</u> prior to the complete display of the web page.

Because the application terminates after rendering the graphical object, it is clear that a terminated rendering application is incapable of providing the claimed "interactive processing," as discussed *supra*.

With respect to the cited Toye reference, the Examiner has considered Professor Felton's affidavit ("Felton II" at paragraph 38) supporting the Patent Owner's contention that Toye teaches away from the proposed combination because existing editor applications at the time of the '906 invention were designed to run in their own dedicated windows.

Whether Toye teaches away with respect to the superimposed display of the X-server output within the "NoteMail" viewer is a close question [see Toye, p. 40, 2nd paragraph, i.e., "any application that displays through an X-server can insert its output (audio, video, or graphics) dynamically onto a notebook page through an embedded 'virtual window.' " ]. The question turns upon whether the Toye "NoteMail" viewing system is equivalent to the browser claimed in the '906 patent and also whether the "embedded virtual window" disclosed by Toye is equivalent to displaying an object within a display area of the "browser-controlled window" claimed in the '906 patent [claims 1 & 6].

As disclosed by Toye, the "NoteMail" system is a hybrid tool that combines "the functions of an engineering notebook, hypermedia browser and authoring environment, mail tool, and file application manager" [Toye, p. 40, col. 1, 3rd paragraph]. With respect to the first prong (i.e., whether the Toye "NoteMail" viewing system is equivalent to the browser claimed in the '906 patent) reasonable arguments may be proffered on both sides.

One might reasonably conclude that the Toye "NoteMail" system is a specialized hypermedia browser, i.e., a species of the genus of hypermedia browsers. "A generic claim cannot be allowed to an applicant if the prior art discloses a species falling within the claimed genus." The species in that case will anticipate the genus. In re Slayter, 276 F.2d 408, 411, 125 USPQ 345, 347 (CCPA 1960); In re Gosteli, 872 F.2d 1008, 10 USPQ2d 1614 (Fed. Cir. 1989).

On the other hand, if the engineering notebook, authoring environment, mail tool, and file application manager functions are the dominant functions of the Toye "NoteMail" viewer, then one could reasonably argue that Toye does not teach the hypermedia browser required by the instant '906 claims when the claims are properly interpreted by applying the broadest reasonable interpretation consistent with the specification. However, the issue of how the instant '906 "hypermedia browser" is construed is not dispositive.

The second prong of inquiry is also a close question, i.e., whether the "embedded virtual window" disclosed by Toye is equivalent to displaying an object at a first location within the display area of the "browser-controlled window" as claimed in the '906 patent. Toye provides further insight regarding the implementation of the "embedded virtual window" by explicitly citing the MediaMosaic article [see Toye, p. 40, col. 2, 3rd paragraph, i.e., "We are aware of only one other multimedia editor with such an architecture, MediaMosaic (22)"].

While Professor Felton's affidavit is technically correct in asserting that existing editor applications at the time of the '906 invention were designed to run in their own dedicated windows, the "virtual window" system disclosed in the MediaMosaic article provides further implementation details. Accordingly, MediaMosaic has been considered by the Examiner as extrinsic evidence to aid in the interpretation of the cited Toye reference. [1]

Felton II at 38 argues:

> 38. Indeed, Toye teaches the use of external editor programs that have not been modified from their standard versions. (See, e.g., Toye at p. 40, col. 2, first full paragraph: "any application that displays through an X-server") Such unmodified programs are not suitable for use within an enclosing document display, because the unmodified programs conventionally display menus and button bars at the top, and other graphical elements around their edges. External application windows with these elements on their borders cannot naturally be displayed within a document display; at most they could be displayed in a window area elsewhere in a windowing environment, as discussed in the previous paragraph. To enable a reasonable editing experience within a document display, the applications would have to be modified; but Toye teaches that they are not modified.

Page 136 of the MediaMosaic article reveals how embedded virtual screens (i.e., embedded virtual windows) were implemented at the time of the Toye reference. MediaMosaic reveals that a virtual screen is a pseudo root window to map X clients so that a portion of their output screens can be embedded in a document as a general media container." Virtual screens use a "pseudo server" that "intercepts and modifies X protocols between X clients and the X server." The protocol essentially "reparents clients to a designated window

[1] Lin, J.K., "MediaMosaic – A Multimedia Editing Environment", Proc. 5th Annual Symposium on User Interface Software and Technology, Monterey CA, Nov. 15-18, 1992 (published by ACM Press).

instead of the root window of the real screen." MediaMosaic creates a virtual screen for a media client and embeds it in a document. MediaMosaic further creates a user-movable and resizable "Viewport" (X Window) for each embedded virtual screen.

The embedded virtual screen is mapped to its corresponding "Viewport" before it is inserted into a document. Text in the document is automatically reformatted around the inserted media displayed within the "Viewport." Significantly, "The mechanism used by MediaMosaic to contain general media is to directly embed them in documents <u>by their original displaying tools</u>" [MediaMosaic, p. 138, 1st paragraph under "5 Duplicated and Full Views].

It is reasonable to assume that Toye uses the MediaMosiac "virtual screen" embedding method because Toye explicitly states that MediaMosaic has the same architecture (i.e., as "NoteMail") [see Toye, p. 40, col. 2, §2].

Prof. Felton's assertion that the applications would have to be modified "because the unmodified programs conventionally display menus and button bars at the top, and other graphical elements around their edges" [see Felton II at 38] is contravened by the extrinsic evidence that MediaMosaic uses the original unmodified rendering tools (i.e., the associated editing applications) to directly embed output media in documents. MediaMosiac simply redirects a portion of the application display output (containing the object to be embedded) to a "virtual screen" associated with a mapped "Viewport."

MediaMosiac appears to operate by cropping out the portion of the application display output that contains the aforementioned display menus, button bars, and other graphical elements around the edges that are normally displayed in the full screen mode of an editing program. Only the embedded object of interest is displayed within the virtual screen associated with the mapped "Viewport" and no modification of the rendering editing application appears to be required [e.g., see Fig. 4, p 139].

MediaMosaic provides an alternate user-selectable full view mode for editing embedded media, as manual resizing of a Viewport window is an awkward way to access the full controls of an associated editing application [see Fig. 5, p. 139].

MediaMosaic therefore provides a mechanism to allow users to embed data objects displayed by different editing applications into one document. Significantly, the system disclosed by MediaMosaic provides the capability to "tailor" (i.e., edit or control) the individual embedded data objects <u>by direct manipulation</u> [MediaMosaic, p. 140, 1st col., §2].

MediaMosaic does enable interactive control and manipulation of objects embedded in what arguably may be construed to be a "browser-controlled window," BUT ONLY AFTER USER INTERVENTION, such as by making a selection with a mouse.

MediaMosaic explicitly discloses: "<u>users</u> can switch media modes <u>by selecting</u> 'Full-View Editing' or 'Embedded-View Editing' from the pull-down menu."

Likewise, Toye teaches that interactive processing is enabled only after a user manually clicks on the "static snapshot" image to launch an external editor program, as discussed *supra.*

Significantly, the prior art approaches of both Toye and MediaMosaic require user intervention to launch an executable application to enable interactive processing. In contrast, the instant '906 claims do not require user intervention to launch the executable application that enables the claimed "interactive processing." Accordingly, for at least this reason, Toye does not anticipate nor render obvious the instant '906 invention.

> *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 6.*

In response, the Examiner finds the Patent Owner's argument I(c) persuasive for at least the following reasons:

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. In re Royka, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." In re Wilson, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970).

The Examiner concurs with the Patent Owner's argument that the proposed five-way combination of references set forth in the last office action does not show automatic invocation of the executable application that enables interactive processing when the hypermedia document is parsed, as claimed.

As persuasively argued by the Patent Owner, the proposed five-way combination of references 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 cited four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II teaches that external data is rendered to a static bit map that is displayed by the browser.

In contrast, Toye teaches that external data is displayed as a "static snapshot" (i.e., representing a data object) within a NoteMail page <u>that must be selected by a mouse to launch an editor application in a separate window</u>" [see Felten II, at paragraph 47]. Thus, Toye clearly requires user intervention to enable interactive processing.

For the aforementioned reasons, the Examiner agrees that all claim limitations are not taught nor fairly suggested by the combination of cited references. Accordingly, the combination proposed in the rejection does not include all the limitations of claims 1 and 6 and a *prima facie* case of obviousness has not been established.

> *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.*

In response, the Examiner finds the Patent Owner's argument II(a) persuasive for at least the following reasons:

As noted *supra*, the previous Examiner of record admitted in making the rejection under 35 U.S.C. §103 of independent claims 1 and 6 that the cited four-way combination of the patent owner's admitted prior art (APA), 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, with distributed hypermedia documents"* [see Office Action mailed Oct. 16, 2004, page 6, lines 18-21].

The Examiner concurs with the Patent Owner's argument that the addition of the Toye reference is a contradiction, and therefore teaches away, from the four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II, because, as the Patent Owner points out, combining Toye with aforementioned four-way combination "would change the principle of operation of the Mosaic, Berners-Lee, Raggett I 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" [see Patent Owner's response, Oct. 12, 2004, page 15, last paragraph].

For at least the aforementioned reason, the cited Toye reference teaches away from the four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II.

> *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.*

In response, the Examiner finds the Patent Owner's argument II(b) persuasive for at least the following reasons:

The Patent Owner points out that "the Mosaic [APA], 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 [see Patent Owner's response, Oct. 12, 2004, page 16, third paragraph from the bottom of the page].

The Patent Owner further observes: "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" [see Patent Owner's response, Oct. 12, 2004, page 16, second from last paragraph].

The Patent Owner further concludes that "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" [see Patent Owner's response, Oct. 12, 2004, page 17, second from last paragraph].

The Examiner concurs with the Patent Owner that the centralized collaborative access system disclosed by Toye teaches away from the distributed system that allows objects to be stored anywhere on the Internet, as taught by the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II. The Examiner agrees that the centralized database approach of Toye has no applicability to the distributed system of the cited Mosaic (APA), Berners-Lee, Raggett I and II combination, and thus Toye teaches away from the four-way combination. A *prima facie* case of obviousness may be rebutted by showing that the art, in any material respect, teaches away from the claimed invention. In re Geisler, 116 F.3d 1465, 1471, 43 USPQ2d 1362, 1366 (Fed. Cir. 1997).

The Examiner finds that the proposed modification would render the prior art invention being modified (i.e., the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II) unsatisfactory for its intended purpose if combined with Toye. The purpose of the Toye centralized collaborative database (i.e., "a collaborative tool for creating, viewing, and sharing multimedia engineering documents in a network environment", see Toye p. 40, col. 1) is distinctly different than the purpose of the cited four-way combination browser that can access another document located anywhere on the Internet.

In contrast, Toye explicitly discloses: "Applications can now reside anywhere on the Internet" as opposed to accessing documents located anywhere on the Internet, as taught by the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II [see Toye, p. 40, col. 2, 2nd paragraph, last line].

Accordingly, Toye teaches away from the cited four-way combination by rendering it unsatisfactory for its intended purpose. If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification. In re Gordon, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984).

Because the system of Toye relies upon a centralized collaborative database as a fundamental principle of operation, and the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II teaches the use of a distributed system that allows objects to be stored anywhere on the Internet, the proposed modification by Toye of the prior art (i.e., Mosaic (APA), Berners-Lee, Raggett I and II) would clearly change the principle of operation of the prior art invention being modified. If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious. In re Ratti, 270 F.2d 810, 123 USPQ 349 (CCPA 1959).

> *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.*

In response, the Examiner finds the Patent Owner's argument II(c) persuasive for at least the following reasons:

As pointed out by the Patent Owner, the Toye reference teaches a system that is appropriate for a collaborative workgroup where the participants know and trust each other and where all authorized users may access and modify the collaborative document after its creation.

The Examiner concurs with the Patent Owner's argument that the publish-once/view-many paradigm that preserves the data and referential integrity (i.e., unidirectional links) defined by the web document author (i.e., as taught by the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II) is destroyed by the modification suggested by the Toye reference.

The addition of the Toye reference clearly teaches away from the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II, because Toye renders the prior art invention being modified unsatisfactory for its intended purpose of preserving the data and referential integrity (i.e., unidirectional links) defined by the web document author. If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification. In re Gordon, 733 F.2d at 900.

> *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.*

As disclosed by Toye, NoteMail "combines the functions of an engineering notebook, hypermedia browser, and authoring environment, mail tool, and file application manager" [see Toye, p. 40, col. 1].

Toye implements a Distributed Information Service (DIS) that Toye defines as follows:

> Conceptually, DIS provides <u>a centralized information storage and management service for all the data associated with a design</u>: 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 server as a reference pointer or "handle" [see Toye, page 40, 2nd column, 2nd from last paragraph].

However, the Patent Owner argues:

> 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." ['906 at col. 5, lines 25-38].

The Felton II affidavit further argues:

> Toye does not teach the use of a 'distributed hypermedia environment,' as that term is used in the '906 claims. The environment provided by Toye is not distributed in the sense of the '906 claims, since it relies on the centralization of a user's document storage in one place. Toye teaches away from the use of a distributed hypermedia environment." (see Felton II, paragraph 25).

The above characterization in Felton II (i.e., "Toye teaches away from the use of a distributed hypermedia environment") is somewhat counterintuitive because Toye teaches the use of combined functions that explicitly include the functions of a "hypermedia browser," and Toye also uses the term "Distributed" in labeling the "Distributed Information Service" [see Toye, p. 40, col. 2].

It appears that the moniker "Distributed" may have been used in labeling Toye's "Distributed Information Service" because centralized information and management services may be <u>distributed</u> to users, e.g., via "persistent objects" that are created in DIS to serve as a reference pointers or handles [see Toye, p. 40, col. 2, 2nd from last paragraph].

The Examiner does not agree with the Patent Owner's assertion that "NoteMail" pages are "not analogous" to Web-style hypermedia documents [see p. 21, 4th paragraph].

Toye explicitly discloses that NoteMail "combines the functions of an engineering notebook, hypermedia browser, and authoring environment, mail tool, and file application manager" [see Toye, p. 40, col. 1].

Toye explicitly discloses the use of "hyper-documents" in the context of an "Internet-wide information web":

> Messages are inserted in chronological order as pages in an electronic design notebook". These pages can be marked up and annotated; items of information can be linked to related items on other pages. The result is a personal hyper-document that captures and structures an engineer's knowledge about a project. Selected information can be shared by e-mailing pages to other engineers or to a central project repository, complete with embedded reference pointers and hyper-links. What emerges is an Internet-wide information web that documents and organizes the shared understanding of an entire engineering team [Toye, p. 40, col. 1].

While it is clear that Toye's spatial arrangement of information items on the "NoteMail" page is implemented with a new "Format" data type [e.g., see Toye, p. 40, col. 2, last paragraph], and is therefore different than the prior art Mosaic (APA), Berners-Lee, Raggett I and II combination, the Examiner does not agree with the Patent Owner's sweeping statement that "NoteMail" pages are not even analogous to Web-style hypermedia documents.

However, the Examiner does find the Patent Owner's final argument to be persuasive and dispositive regarding argument III(a):

> 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 11, at paragraphs 36-37].

The Examiner concurs because 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 the data. Therefore, Toye teaches away from the operative coupling between the "executable application" and the "interactive processing" required by the instant '906 patent claims.

Furthermore, Toye teaches that "automatic invoking" of the "appropriate application" is performed by <u>selection</u>, and not by parsing. Toye teaches that notebook data is displayed as a data object or filename <u>that must be selected by a mouse to launch an appropriate application in a separate window</u>" [see Toye page 40, 2nd column, paragraph 2; see also page 36, 2nd column, last paragraph, i.e., " ... ability to construct hyper-documents containing <u>bitmaps</u>, video, and audio"; see also Felten II, at paragraph 47].

Significantly, Toye appears to merely disclose a conventional system for invoking appropriate applications by standard prior art file association techniques, such as invoking the appropriate application based upon the file extension (e.g., when the user clicks and selects a *.doc filename or corresponding file icon and this user action automatically invokes the appropriate word processor). See also Toye: "The functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file" [p. 40, 2nd column, 2nd paragraph].

*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 response, the Examiner finds the Patent Owner's argument III(b) persuasive for at least the following reasons:

The Examiner notes that the term "dynamic object" is not explicitly used in the Toye disclosure, nor is the term used within the instant '906 claims. It appears the previous Examiner is interpreting the Toye reference to teach the use of an embedded object that is dynamic in the sense that the embedded object may be interactively changed by the user while it is being displayed.

The Examiner concurs and finds dispositive the Patent Owner's argument that the "dynamic objects" taught by Toye are "activated by the user clicking on a static "snap shot" image or icon displayed within a NoteMail page" [see Patent Owner's response, received Oct. 12, 2004, p 22].

The Examiner concurs that the link between the "dynamic object" allegedly taught by Toye and the application to process the "dynamic object" is stored in an external centralized database, and not within the "NoteMail" page itself (as contrasted with the use of the EMBED tag disclosed by Raggett I that provides the link to a rendering application, discussed *supra*; see Raggett I, p. 6).

Accordingly, Toye fails to teach or fairly suggest an ongoing real-time modification or control by a user of a displayed object shown within a browser-controlled window, as performed by an "executable application" that is invoked by parsing an "EMBED" tag to enable "interactive processing" of the type claimed in the '906 patent.

*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.*

In response, the Examiner finds the Patent Owner's argument IV(a) persuasive for at least the following reasons:

"In determining the propriety of the Patent Office case for obviousness in the first instance, it is necessary to ascertain whether or not the reference teachings would appear to be sufficient for one of ordinary skill in the relevant art having the reference before him to make the proposed substitution, combination, or other modification." <u>In re Linter</u>, 458 F.2d 1013, 1016, 173 USPQ 560, 562 (CCPA 1972). The mere fact that references <u>can</u> be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. <u>In re Mills</u>, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990).

In the rejection set forth on page 6 of the Office Action mailed Oct. 16, 2004, the previous Examiner asserts that the modification of the four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II, would be motivated based upon "Toye's teaching that its architecture provides openness and flexibility":

> The combination of patentee'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, with distributed hypermedia documents.
>
> 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 (See Toye: p. 40 description of NoteMail, particularly p. 40, col. 2, first paragraph).
>
> It would have been readily apparent to a skilled artisan to modify the method discussed above, combining the teachings of the admitted prior ad in view of Berners-Lee, Raggett I, and Raggett II, by further modifying the combination's static embedded object to be a dynamic embedded object as taught by Toye. **Such a further modification would have been apparent based on Toye's teaching that its architecture provides openness and flexibility** (See Toye: p. 40 col. 2 second complete paragraph).

The support for the "openness and flexibility" motivation relied upon the previous Examiner is taken from the following section of the Toye reference [see p. 40, 2nd column, 2nd and 3rd complete paragraphs]:

> Another 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 onto a notebook page through an embedded "virtual window". When a data object or file is **selected** for inclusion in the

> notebook, the system **will automatically invoke the appropriate application** for displaying that item in the notebook. If the needed application is not locally resident (a likely occurrence in the case of MIME external body references), it will be located and run remotely over the network. Subsequently selecting the displayed data with a mouse will restart the original application, so that the data can be edited or updated without leaving the notebook environment. **The functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file.** However, applications can now reside anywhere on the Internet.
>
> We are aware of only one other multimedia editor with such an architecture, MediaMosaic [22]. Other engineering notebook projects, by contrast lack this **openness and flexibility**. For example, the Virtual Notebook System [6] can display only static bitmaps; GE's Electronic design Notebook [34], which is built on FrameMaker, can run only those applications whose output formats are compatible with the handful of input formats that FrameMaker accepts.

The Patent Owner argues: "the general and nebulous Toye language regarding 'openness and flexibility' is not related to any possible motivation to combine the references" [see Patent Owner's response received Oct. 12, 2004, p. 23].

In response, the Examiner concurs with the Patent Owner's contention that the "openness and flexibility" motivation applied by the previous Examiner is general and nebulous, for the following reasons:

"Openness and flexibility" is supported by the term "open architecture" in paragraph 2, *supra*, describing a "virtual window" for displaying the output of <u>any application</u> (i.e., suggesting "flexibility") that can display its output through an X-Server. As disclosed by Toye, "any application that displays through an X-server can insert its output (audio, video, or graphics) dynamically onto a notebook page through an embedded 'virtual window' [see Toye, p. 40, 2nd column, paragraph 2]. Toye also teaches a "flexible" system in the sense that if a needed application is not locally resident, it will be "located and run remotely over the network" [see Toye, p. 40, 2nd column, 2nd paragraph].

With respect to the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II, it is conceded that the section of Toye cited by the previous Examiner would likely provide a motivation to a skilled artisan to modify the four-way combination for the purpose of making it compatible, e.g., with applications that display through an X-Window system, using an X-server.

It is also conceded that, <u>after a user makes a manual selection of a "data object or file</u>," Toye teaches that a local or remote editing application is invoked that can display dynamic objects such as audio and video that may be displayed as an embedded object within a notebook page using the disclosed "virtual window."

However, there is no suggestion to modify the four-way combination to allow a user to interactively process an object embedded within a distributed hypermedia document in accordance with the type of "interactive processing" recited in claims 1 and 6 of the instant '906 patent.

While Toye certainly teaches that the user may select a data object or file and "automatically invoke the appropriate application for displaying that item in the notebook" (as typically performed using file associations in a conventional file manager program) such interactivity (as taught by Toye) <u>can only be initiated by a manual selection performed by the user</u> (i.e., a mouse click or other user selection, as by using a keyboard).

The manual selection step required by Toye defeats the purpose of the use of an EMBED tag that is parsed to invoke an executable application, thus teaching away from the hypothetical four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II.

In contrast, the instant '906 claims require the browser (and not the user) to invoke the "executable application" that in turn executes on the client workstation to enable the claimed "interactive processing."

Accordingly, the Toye reference teaching is insufficient to enable one of ordinary skill in the relevant art having the reference before him to make the proposed substitution, combination, or other modification.

*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.*

"To support the conclusion that the claimed invention is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed invention or the examiner must present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references." Ex parte Clapp, 227 USPQ 972, 973 (Bd. Pat. App. & Inter. 1985).

The Mosaic (APA), Berners-Lee, Raggett I and II combination provides a distributed system where objects may be stored anywhere on the Internet and retrieved using a browser application, e.g., by 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 that uses a single object-oriented database (DIS) to access and store documents.

The Examiner finds the Patent Owner's argument compelling that "the collaborative editing techniques of Toye would be contrary to the publish-and-view philosophy of the Internet." Furthermore, the Examiner concurs that "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" [see Patent Owner's response received Oct. 12, 2004, page 23].

The five-way rejection set forth in the last office action (including the Toye reference) fails to provide a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references.

While Toye does teach dynamic objects (such as audio and video) that may be displayed within the same notebook window using an overlay "virtual window" X-Windows technique, the interactive processing (i.e., editing) taught by Toye <u>can only be invoked manual selection of a data object or file by a user</u> and is therefore not equivalent to the type of interactive processing claimed by the instant '906 patent.

In contrast, the instant '906 claims require the browser (not the user) to invoke the "executable application" that in turn executes on the client workstation to enable the claimed "interactive processing."

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

The "NoteMail" tool combines the functions of an engineering notebook, hypermedia browser and authoring environment, and a file application manager [see Toye, p. 40, col. 1]. The "NoteMail" system is organized in a manner designed to provide maximum benefit to members of a collaborative engineering team.

For example, messages are inserted in <u>chronological order</u> as "NoteMail" pages in an approach that departs from the functionality of prior art web browsers. Prior art web browsers typically organize web page retrieval around stored bookmarks that provide URL links to web pages (and associated objects) that may reside anywhere on the Internet.

The Examiner concurs with the Patent Owner's contention that the "NoteMail" design (i.e., teaching restricted, collaborative access to a centralized database) runs counter to the intended purpose of the Mosaic (APA), Berners-Lee, Raggett I and II hypothetical four-way combination. The intended purpose of the four-way combination is to provide a distributed system that enables universal access to web pages (and associated objects) that may be stored anywhere on the Internet.

In contrast, Toye discloses a system that permits <u>applications</u> to reside anywhere on the Internet, while collaborative, restricted access to the data is only permitted via a centralized database [Toye, p. 40, col. 2, paragraph 2, last line].

Accordingly, the Examiner concurs that the Toye reference teaches away from modifying the Mosaic (APA), Berners-Lee, Raggett I and II hypothetical four-way combination as proposed by the rejection set forth in the last office action.

*PART V. The secondary consideration of commercial success further supports the conclusion of non-obviousness. The attached Declaration of Robert J. Dolan, Dean at the University of Michigan Business School and Gilbert and Ruth Whitaker professor at Michigan Business School ("Dolan") sets forth facts and evidence to legally and factually establish the secondary consideration of commercial success of the invention claimed in claims 1 and 6 of the '906 patent.*

*a. There is a nexus between the claimed invention and the commercial success.*

In response to the Patent Owner's argument V(a), the Examiner has reviewed the supporting "Dolan" Declaration and does not find it persuasive in terms of demonstrating a nexus between the instant claimed '906 invention and commercial success.

The "Dolan" Declaration relies upon the alleged infringement of the '906 patent claims by Microsoft in marketing the Microsoft Internet Explorer browser (IE). In particular, it is alleged that the "IE browser's support for "plug-ins, applets, and Active X functionality incorporates the technology claimed in claims 1 and 6 of the '906 patent" [See "Dolan" Declaration, page 2].

The Patent Owner's argument of commercial success is thus predicated on Microsoft's infringement of the '906 patent as determined by a jury in the trial at the U.S. District Court (Northern District of Illinois, Eastern Division).

However, at the time of this writing, the litigation is still ongoing and is on remand back to the District Court from the CAFC.

While the infringement issue is not being considered on remand from the CAFC, the affirmative defenses of public use and inequitable conduct, if successful, would render the patent invalid and the issue of patent infringement would be moot. Therefore, the PTO does not consider there to be a final judgment on the issue of patent infringement until all appeals have been exhausted and the litigation has concluded. A nexus between the claimed invention and the commercial success of the IE browser cannot be shown (based upon alleged patent infringement) in the absence of a final judgment to establish such infringement.

Accordingly, the Patent Owner has not met the burden of proof required to establish a factual and legally sufficient connection between the evidence of commercial success and the claimed invention such that the evidence is of probative value in the determination of nonobviousness.

*b. The evidence of commercial success is commensurate with the scope of the '906 claims.*

Objective evidence of nonobviousness including commercial success must be commensurate in scope with the claims. In re Tiffin, 448 F.2d 791, 171 USPQ 294 (CCPA 1971). In order to be commensurate in scope with the claims, the commercial success must be due to claimed features, and not due to unclaimed features. Joy Technologies Inc. v. Manbeck, 751 F. Supp.

225, 229, 17 USPQ2d 1257, 1260 (D.D.C. 1990), *aff'd*, 959 F.2d 226, 228, 22 USPQ2d 1153, 1156 (Fed. Cir. 1992).

The Patent Owner relies upon the "Dolan" Declaration (pages 6-9, numbered paragraphs 25-41) to support the contention that the evidence of commercial success is commensurate in scope with claims 1 and 6 of the instant '906 patent.

In response to the Patent Owner's argument V(b), the Examiner need not reach this issue because a nexus between the claimed invention and commercial success has not been established, as discussed in the response to argument V(a), *supra.* A nexus between the claimed invention and the commercial success of the IE browser cannot be shown (based upon alleged patent infringement) in the absence of a final court judgment to establish such infringement (i.e., until all appeals have been exhausted and the litigation has concluded).

The Examiner cannot reasonably address the issue raised by argument V(b) without commenting on the merits of the ongoing litigation. Subject matter concerning patent infringement constitutes a federal question that properly falls within the subject matter jurisdiction of the Federal Court system. Subject matter concerning patent infringement is not considered by the U.S. Patent and Trademark Office.

*c. The commercial success is derived from the invention.*

In response to the Patent Owner's argument V(c), the Examiner does not find the Patent Owner's arguments and the associated "Dolan" Declaration persuasive for the following reasons:

In considering evidence of commercial success, care should be taken to determine that the commercial success alleged is directly derived from the invention claimed, in a marketplace where the consumer is free to choose on the basis of objective principles, and that such success <u>is not the result of heavy promotion or advertising</u>, shift in advertising, consumption by purchasers normally tied to applicant or assignee, or other business events extraneous to the merits of the claimed invention, etc. <u>In re Mageli</u>, 470 F.2d 1380, 176 USPQ 305 (CCPA 1973) (conclusory statements or opinions that increased sales were due to the merits of the invention are entitled to little weight); <u>In re Noznick</u>, 478 F.2d 1260, 178 USPQ 43 (CCPA 1973).

Even assuming, arguendo, that the Patent Owner has demonstrated the required nexus between the instant claimed '906 invention and the commercial success of an allegedly infringing product, the "Dolan" Declaration fails to show that the commercial success of Microsoft's IE browser was not the result of <u>heavy promotion or advertising</u> or other business events extraneous to the merits of the claimed invention.

In particular, Microsoft made the IE browser available to users at little or no cost. Microsoft also bundled the IE browser as an integral component of various Microsoft operating systems (e.g., Windows 95, 98, and Windows

2000). Significantly, the "Dolan" Declaration is silent regarding the issue of free or low cost distribution of the IE browser as a factor in Microsoft's successful capture of market share.

In more traditional business models that involve tangible products, a rational producer will seek to exploit a profit opportunity until the marginal cost of the nth unit produced exceeds the marginal revenue generated from that nth unit. However, when software is distributed over the Internet, the marginal cost of each unit of downloaded software approaches zero as the number of downloads approaches infinity. This is true because the sunk software development costs and the relatively fixed cost of maintaining distribution servers are averaged over a potentially infinite number of downloads.

Obviously, if there exists a quantifiable market demand for a given product, the quantity of units demanded will increase as the cost per unit approaches zero. This was likely true in the case of the Microsoft IE browser because it was offered to the public as a free download (or merely for the cost of the CD media plus postage and handling).

Microsoft clearly offered the IE browser to the public at little or no cost in an effort to gain market share over the competing Netscape browser, even though it may also be true that Microsoft viewed the functionality of Active X (allegedly infringing upon the '906 patent functionality) as giving IE an advantage over Netscape [e.g., see "Dolan" Declaration, page 8, paragraph 36]. In addition, such free distribution of the IE browser clearly promoted and helped to advertise Microsoft's main operating system and application software products.

Because Microsoft made the IE browser available to the public at little or no cost, the past distribution of IE has at least the appearance of "heavy promotion or advertising." While the alleged infringement of '906 functionality may indeed have been a factor in the market success of the IE browser, patent infringement has not been shown by a final court judgment. Significantly, the Patent Owner has failed to address the Microsoft marketing strategy of distributing the IE browser to the public at little or no cost.

Because Microsoft was already an established market leader with respect to desktop operating systems and applications, the success of the IE browser could also be reasonably attributed to Microsoft's extensive advertising and position as a market leader before the introduction of the allegedly infringing product (i.e., the IE browser). See Pentec, Inc. v. Graphic Controls Corp., 776 F.2d 309, 227 USPQ 766 (Fed. Cir. 1985) (commercial success may have been attributable to extensive advertising and position as a market leader before the introduction of the patented product).

Accordingly, even when the facts are viewed in a light most favorable to the Patent Owner, the Patent Owner has failed to demonstrate by a preponderance of the evidence that the commercial success of Microsoft's IE browser was derived from the instant '906 invention.

## The Viola Code

The CAFC opinion (Docket No. 04-1234, March 2, 2005) states on page 11, 2nd paragraph:

> In contrast, the record indicates Wei not only demonstrated DX34 to two Sun Microsystems engineers without a confidentiality agreement (on May 7, 1993), but only twenty-four days later (on May 31, 1993) posted DX37 on a publicly-accessible Internet site and notified a Sun Microsystems engineer that DX37 was available for downloading.

The '906 invention was reduced to practice no later than January, 27, 1994 when it was presented on that date to a conference "Medicine Meets Virtual Reality II." [2] From the court record, it is clear that the date of publication on the Internet of the DX37 code (May 31, 1993) antedates the date of reduction to practice (Jan. 27, 1994) of the '906 invention. Accordingly, the DX37 code submitted by the Patent Owner on Dec. 30, 2003 (received by the PTO on Jan 5, 2004) has been considered by the Patent and Trademark Office as a publication that constitutes prior art for purposes of this reexamination proceeding.

The "Viola Code" is stored as an artifact (i.e., a CD disk) associated with the instant Image File Wrapper (IFW) reexamination file. The contents of artifacts are not stored as images on the PTO IFW system. The Viola code CD contains two compressed zip files representing "Viola Source code" ("DX34" and "DX37"):

[2] See "Ruling on the Defense of Inequitable Conduct", No. 99 C 626, U.S. District Court Northern District of Illinois, Eastern Division, page 9.

1) viola930512.tar.gz.zip - this compressed file represents the earlier Viola source code, also referred to as "DX34" in the CAFC opinion (Docket no. 04-1234, March 2, 2005, see also IFW "Reexam Notice of Court Action" dated April 11, 2005; see especially page 11 as numbered in the printout (corresponding to IFW page 16 of 32). The viola930512.tar.gz.zip (i.e., "DX34") file, when unzipped, contains 1,027 files in 35 folders consisting of 8 total megabytes in size.

2) violaTOGO.tar.Z.zip - this compressed file represents the later Viola source code, also referred to as "DX37" in the CAFC opinion. The violaTOGO.tar.Z.zip (i.e., "DX37") file, when unzipped, contains 1,030 files in 34 folders consisting of 7.7 total megabytes in size.

To conduct a thorough and comprehensive review of the DX37 code (1,030 files), the Examiner successfully unzipped the provided "violaTOGO.tar.Z.zip" compressed file and indexed all DX37 files using a commercially available text searching program designed for such purpose. [3]

In this manner, every DX37 file containing textual content (including code) was fully and comprehensively text searched with the resulting "hits" being highlighted in the full-text context of each document. Several representative Viola files are reproduced *infra* to clarify the scope of the Viola DX37 prior art publication.

[3] The Examiner used the "dtSearch" program to index and text search all DX7 files that contained textual content. See **http://www.dtsearch.com/**

**How Viola embeds Viola scripts in a hypermedia document**

In particular, the file "violaApps.hmml" (contained in the "docs" directory) illustrates how interactive applications (i.e., actually Viola scripts) are embedded in a Viola hypermedia document as designated by a matched pair of <VOBJF> and </VOBJF> tags that specify a Viola script that is used to generate the embedded object, as shown below:

```
<VOBJF> ../apps/clock.v </VOBJF>
```

When the Viola hypermedia browser parses the hypermedia document (e.g., "violaApps.hmml", denoting a hypermedia document written in Hyper Media Markup Language) and encounters the matched pair <VOBJF> and </VOBJF> tags, the browser then retrieves the Viola script "clock.v" from the directory location specified by the directory path (i.e., ../apps/ ).

Significantly, the Viola script "clock.v" is INTERPRETED to embed an interactive application object within the same window of the Viola browser. Each Viola script line is interpreted by translating the Viola script code (or corresponding byte code) to native binary machine code instructions that are executed in a sequential fashion.

The Viola documentation states: "The extension language is C-like in syntax and is processed into byte-code for efficient interpretation" [see "violaCh1.hmml" in the "docs" directory]. Although the aforementioned "clock.v" example is clearly a Viola script, it appears that an intermediate byte-code representation may be interpreted at runtime. In such case, the Viola script must be compiled in advance to intermediate byte-code form.

The "violaApps.hmml" hypermedia document file (as parsed by the Viola browser) and the corresponding "clock.v" script file are shown below:

"violaApps.hmml" illustrating the use of the Viola <VOBJF> object tags (located within the "docs" directory)

```
<!DOCTYPE hmml SYSTEM>
<TITLE>Test</TITLE>
<H1>List No. 5</H1>
<P>
The <CMD>&lt;VOBJF&gt;</CMD> tag can be used to insert viola
applications.
Using this capability allows you embed in your document what you
can access or build using viola's programming, and GUIs. Of
course too much violaism reduces the portability of your document
on the World Wide Web,but anyway...
<P>
Here are some examples.
<H2>Clock</H2>
<VOBJF>../apps/clock.v</VOBJF>
<H2>Vicon</H2>
<VOBJF>../apps/vicon.v</VOBJF>
<P>
This can be a handy menu to tuck away at a corner of the screen.
<H2>Query</H2>
<VOBJF>../apps/vwq.v</VOBJF>
<P>
This application is intended to gather user information.
<H2>Wave fun</H2>
<VOBJF>../apps/wave.v</VOBJF>
<H2>Noodle Doodles</H2>
<VOBJF>../apps/doodle.v</VOBJF>
<P>
So I was bored...
<P>
The end.
```

The first portion of the corresponding "clock.v" Viola script (located in the "apps" directory)

```
\name {clock}
\class {vpane}
\parent {}
\width {200}
\height {210}
\children {clock.dial clock.mesg}
```

```
\
\name {clock.dial}
\class {XPMBG}
\parent {clock}
\script     {
      print("@@@@@@ clock: ");
      for (i =0; i < arg[]; i++) print(arg[i], ", ");
      print("\n");

      switch (arg[0]) {
      case "tick":

            date = date();
            clock.mesg("update");
            second = int(nthWord(date, 6));
            minute = int(nthWord(date, 5));
            hour = int(nthWord(date, 4));
            if (hour >= 12) hour = hour - 12;

            secondD = (second / 60.0 * 360.0) - 90.0;
            minuteD = (minute / 60.0 * 360.0) - 90.0;
            hourD = (hour / 12.0 * 360.0) - 90.0 + (minute / 60.0 * 30.0);

            secondX = secondR * cos(secondD) + centerX;
            secondY = secondR * sin(secondD) + centerY;
            minuteX = minuteR * cos(minuteD) + centerX;
            minuteY = minuteR * sin(minuteD) + centerY;
            hourX = hourR * cos(hourD) + centerX;
            hourY = hourR * sin(hourD) + centerY;

            if (lminuteX != minuteX) {
                  clearWindow();
                  clock.dial("render"); /* brutally redraw */
                  drawLine(centerX, centerY, minuteX, minuteY);
                  drawLine(centerX, centerY, hourX, hourY);
                  invertLine(centerX, centerY, lsecondX, lsecondY);
            }
            invertLine(centerX, centerY, lsecondX, lsecondY);
            invertLine(centerX, centerY, secondX, secondY);

            lsecondX = secondX;
            lsecondY = secondY;
            lminuteX = minuteX;
            lminuteY = minuteY;
            lhourX = hourX;
            lhourY = hourY;
            if (view) after(1000, "clock.dial", "tick");
            return;
      break;
      case "render":
            usual();
            for (i = 1; i <= 12; i = i + 1) {
                    x = letterR * cos((i / 12.0 * 360) - 90) +
                          centerX - 10;
                    y = letterR * sin((i / 12.0 * 360) - 90) +
                          centerY - 5;
                    drawText(x, y, i, str(i));
              }
              return;
        break;
        case "VIEW_ON":
              view = 1;
              return;
        break;
        case "VIEW_OFF":
              view = 0;
              return;
        break;
        case "expose":
              clearWindow();
              lminuteX = 0; /* forces redrawing */
              lhourX = 0; /* forces redrawing */
        break;
        case "config":
              usual();
              send(self(), "resize", arg[3], arg[4]);
              return;
        break;
        case "resize":
              if (arg[1] < arg[2])
                    radius = arg[1] / 2.0;
              else
                    radius = arg[2] / 2.0;

              centerX = arg[1] / 2.0;
              centerY = arg[2] / 2.0;
              secondR = radius * 0.95;
              minuteR = radius * 0.9;
              hourR       = radius * 0.6;
              letterR = (radius - 9) * 0.94;

              after(2000, "clock.dial", "tick");
              lminuteX = 0;

/*            system(concat(environVar("VIOLA"), "/play ",
                          environVar("VIOLA_DOCS"), "/cuckoo.au"));
*/
        break;
        }
        usual();
}
```

The Viola DX37 approach to embedding interactive objects using interpreted Viola scripts (or corresponding byte-code forms) does not anticipate nor fairly suggest the '906 invention as claimed for at least the following reasons:

While Viola DX37 supports hypermedia and a type of interpreted script-based interactive processing, the Examiner can find no indication from a comprehensive text search of the Viola DX37 files that such interactivity results from the use of a parsed **embed text format** that specifies the location of an **object** external to the hypermedia document, where the browser application **uses type information associated with the object to identify and locate an external executable application**, and where the parsing step results in the browser automatically invoking the **executable application** to display the **object** and enable interactive processing of the **object** within the same browser-controlled window, when the instant '906 patent claims 1 and 6 are properly accorded the broadest reasonable interpretation consistent with the specification.

## I. VIOLA <VOBJF> TAGS DO NOT ANTICIPATE NOR FAIRLY SUGGEST THE "EMBED TEXT FORMAT" AS CLAIMED IN THE '906 PATENT.

Unlike the instant '906 claimed "embed text format," the Viola <VOBJF> tags use no arguments or additional elements beyond a directory path and filename. The Viola <VOBJF> tag simply loads the Viola script using the path and filename specified between the <VOBJF> and </VOBJF> tags, as shown:

```
<VOBJF> ../apps/clock.v </VOBJF>
```

In contrast, the browser application of the instant '906 patent uses a type element associated with the external object (i.e., "type information" as claimed) to identify and locate an executable application external to the distributed hypermedia document [see '906 patent, TABLE II and associated discussion col. 13].

Significantly, the Viola browser application <u>does not fairly teach nor suggest where the browser application uses type information associated with the external object to identify and locate an external executable application</u>.

## II. VIOLA SCRIPTS (OR CORRESPONDING BYTE-CODE FORMS) DO NOT ANTICIPATE NOR FAIRLY SUGGEST THE EXTERNAL "OBJECT" AS CLAIMED IN THE '906 PATENT.

If the Viola <VOBJF> tags are considered as arguably corresponding to the instant claimed '906 "embed text format" (in the sense that the Viola <VOBJF> tags specify "the location of at least a portion of an object external to the first distributed hypermedia document" as claimed in '906 claims 1 and 6), then the Viola script program specified between the <VOBJF> tags is not equivalent to the instant '906 claimed external "object" when the claimed '906 external "object" is interpreted in a manner consistent with the specification of the '906 patent.

The Viola, "clock.v" script is a high-level source code PROGRAM. In contrast, the scope of the claimed '906 external "object" broadly encompasses myriad types of data objects, including self-extracting data objects [see '906 patent, col. 3, lines 33-51].

The scope of the claimed '906 external "object" is broad when construed in a manner consistent with the specification (i.e., see '906 patent, col. 3, lines 36-39: "a data object is information capable of being retrieved and presented to a user of a computer system."). However, the scope of the claimed '906 external "object" clearly does not read upon a high-level source code PROGRAM, such as a Viola script, nor does it read upon an object in byte-code form.

When the scope of the claimed '906 external "object" is construed in a manner consistent with the specification, it is clear that any executable component of the claimed '906 external data "object" is limited to performing self-extraction of the compressed data object:

See '906 patent, col. 3, lines 43-51:

> When a browser retrieves an object such as a self-extracting **data object** the browser may allow the user to "launch" the self-extracting **data object** to automatically execute the unpacking instructions to expand the data object to its original size. Such a combination of executable code and data is limited in that the user can do no more than invoke the code to perform a singular

<u>function such as performing the self-extraction after which time the object is a standard data object</u>.

Although a self-extracting data object typically includes executable code to expand the compressed data object to its original size, this type of self-extraction extracts DATA that has no relationship to a high-level source code PROGRAM in the form of a Viola script, or a byte-code file, or the like.

## III. VIOLA SCRIPTS (OR CORRESPONDING BYTE-CODE FORMS) DO NOT ANTICIPATE NOR FAIRLY SUGGEST THE EXTERNAL "EXECUTABLE APPLICATION" AS CLAIMED IN THE '906 PATENT.

The Examiner finds that the Viola code publication does not fairly teach nor suggest that the browser automatically invokes an **executable application**, external to the hypermedia document, to display the object and enable interactive processing of the object, when the instant '906 patent claims 1 and 6 are properly accorded the broadest reasonable interpretation <u>consistent with the specification</u>, where such interpretation is also <u>consistent with the interpretation that those skilled in the art would reach</u>. <u>In re Hyatt</u>, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000); <u>In re Cortright</u>, 165 F.3d 1353, 1359, 49 USPQ2d 1464, 1468 (Fed. Cir. 1999).

While expert witnesses and dictionaries (considered as extrinsic evidence) may differ regarding the proper construction of the instant claimed "executable application", the Central Processing Unit (i.e., CPU or microprocessor) found in every computer system has only a single, precisely defined interpretation as to what constitutes an "executable application." When the CPU initiates a "fetch and execute" cycle, the program counter is loaded with the address of the next executable instruction. To be "executable" the contents of the memory location pointed to by the program counter must contain an instruction in binary form that is a member of the native instruction set of the microprocessor (i.e., a binary machine language instruction). The binary representation of the precise portion of the machine language instruction that determines what kind of action the computer should take (e.g., add, jump, load, store) is referred to as an operation code (i.e., OP code). From the perspective of the CPU, if a recognizable machine language instruction (i.e., a native CPU instruction) is not found within the memory location pointed to by the program counter, the computer will crash.

The Viola system uses "C-like" Viola scripts that must be INTERPRETED by the browser and then TRANSLATED or CONVERTED into binary native executable machine code that can be understood by the CPU. Alternately, the Viola script is precompiled to intermediate byte-code form and the byte-code is interpreted (i.e., translated) into binary native executable machine code at runtime. This extra step of translation results in an unavoidable performance penalty, as interpreted applications run much slower than compiled native binary executable applications.

Accordingly, the "C-like" Viola scripts (or corresponding byte-code representations) are not "executable applications" from the perspective of the CPU, which is the only perspective that really matters at runtime. A conventional CPU is only capable of processing binary machine language instructions from its own native instruction set.

Without an intermediate translation step performed by an interpreter component of the Viola browser, a Viola script (or corresponding byte-code representation) cannot be processed as an executable application by the CPU.

Significantly, the instant '906 specification is silent regarding the use of applications that rely upon scripts that must be interpreted before they can be executed. The instant '906 specification is silent with respect to interpreting code prior to execution. The instant '906 specification is silent with respect to the use of byte-code intermediate forms.

## IV. THE INTENDED USE OF THE VIOLA RAPID PROTOTYPING INTERPRETED SCRIPTING SYSTEM TEACHES AWAY FROM THE INTENDED USE OF THE '906 PATENT.

The Viola scripting system teaches away from the primary intended use of the '906 invention. The main object of the Viola scripting system was to provide an <u>interpreted</u> operating environment <u>primarily designed for rapid prototyping</u>.

In contrast, the main object of the '906 invention is to provide a system "that allows the accessing, display and manipulation of large amounts of data, especially image data, over the Internet to a small, and relatively cheap, client computer ['906 patent, col. 6, lines 21-25].

The use of an interpreted script application (or corresponding intermediate byte-code representation) in the '906 patent context would be unacceptably slow in processing large amounts of data, especially the kind of complex three-dimensional image data used in one embodiment of the '906 patent. One must reflect on the fact that the personal computers used in 1994 were significantly slower than the high speed computers widely used today (2005).

Overcoming the existing bandwidth and processing speed constraints associated with the prior art are central objects of the '906 invention [see '906 patent, col. 5, lines 39-56]:

> The open distributed hypermedia system provided by the Internet allows users to easily access and retrieve different data objects located in remote geographic locations on the Internet. However, this open distributed hypermedia system as it currently exists **has shortcomings** in that today's large data objects are limited largely by **bandwidth constraints** in the various communication links in the Internet and localized networks, and by the limited processing power, or computing constraints, of small computer systems normally provided to most users. **Large data objects are difficult to update at frame rates fast enough (e.g., 30 frames per second) to achieve smooth animation.** Moreover, **the processing power needed to perform the calculations to animate such images in real time does not exist on most workstations, not to mention personal computers.** Today's browsers and viewers **are not capable of performing the computation necessary to generate and render new views of these large data objects in real time.**

Also see '906 patent, col. 6, lines 21-31:

> On the other hand, small client computers in the form of personal computers or workstations such as client computer 108 of FIG. 2 are generally available to a much larger number of researchers. Further, it is common for these smaller computers to be connected to the Internet. **Thus, it is desirable to have a system that allows the accessing, display and manipulation of large amounts of data, especially**

> **image data, over the Internet to a small, and relatively cheap, client computer.**
>
> Due to the relatively **low bandwidth of the Internet** (as compared to today's large data objects) and the **relatively small amount of processing power available at client computers**, many valuable tasks performed by computers cannot be performed by users at client computers on the Internet.

The importance of "speed of access" to application client 210 (corresponding to the instant claimed "executable application") is further demonstrated by the use of "Terminate and Stay Resident" (TSR) programs to provide faster access [See '906 patent, col. 8, lines 66, 67, cont'd, col. 9, lines 1-14]:

> Client computer 200 includes processes, such as browser client 208 and **application client 210.** In a preferred embodiment, **application client 210** is resident within client computer 200 prior to browser client 208's parsing of a hypermedia document as discussed below. In a preferred embodiment application client 210 resides on the hard disk or RAM of client computer 200 and is loaded (if necessary) and executed when browser client 208 detects a link to **application client 210**. The preferred embodiment uses the XEvent interprocess communication protocol to exchange information between browser client 208 and **application client 210** as described in more detail, below. Another possibility is to install **application client 210** as a **"terminate and stay resident" (TSR) program** in an operating system environment, such as X-Window. Thereby making access to **application client 210** much faster.

The Examiner submits that "Terminate and Stay Resident" (TSR) programs were notoriously understood to be native binary executable code by those of ordinary skill in the art at the time of the '906 invention. [4]

For example, in the legacy Microsoft MS-DOS environment, TSR programs were native binary executables designated as COM or EXE programs that were preloaded in memory for fast execution. TSR programs were typically used to allow utilities, drivers, or interrupt handlers to be preloaded in memory for quick access. [5] The purpose of memory preloading for quick access would not be well served if a TSR program in the form of a script had to be interpreted (i.e., translated) to binary native code before it could be executed.

In addition, the '906 patent teaches the use of applications such as "spreadsheet programs, database programs, and word processor programs" [see col. 13, line 14]. The Examiner submits that at the time of the invention most commercial spreadsheet programs, database programs, and word processor applications were usually sold as native binary executable applications. The Examiner does concede that applications of the aforementioned types were available in interpreted languages at the time of the invention (e.g., a database program written in the BASIC language). However, an interpreted application in source code form cannot be executed directly by the CPU without first being translated to native binary executable machine code form, as discussed *supra.*

[4] See e.g., U.S. Patent 5,056,057 to Johnson et al., "Keyboard interface for use in computers incorporating terminate-and-stay-resident programs", issued Oct. 8, 1991.
[5] Duncan, Ray, "Advanced MSDOS Programming", Microsoft Press, 1986, page 391.

## V. EVEN ASSUMING, ARGUENDO, THAT "INTERPRETING A SCRIPT" (OR CORRESPONDING BYTE-CODE REPRESENTATION) MAY BE BROADLY CONSIDERED AS EQUIVALENT TO "EXECUTING AN APPLICATION", SUCH INTEPRETATION MERGES THE BROWSER AND THE "EXECUTABLE APPLICATION" INTO ONE PROGRAM THAT FAILS TO TEACH EVERY ELEMENT OF THE '906 PATENT CLAIMS.

Assuming *arguendo* that one adopts the alternate broader modern construction where "interpreting a script" (or interpreted the corresponding byte-code representation) may be considered as equivalent to "executing an application," then the Viola script arguably becomes an integral component of the Viola browser that parses, interprets (i.e. translates), and executes each line of the script (or corresponding byte-code). In such case, the browser and the "executable application" merge into one program, and therefore cannot meet the requirement for a discrete "browser application" and a discrete "executable application" as claimed by the instant '906 patent [see claims 1 and 6].

Lastly, The Examiner takes particular note of the fourth line of the "violaBrief.hmml" file ("Technical Overview of Viola," see the "docs" directory) that leads one to conclude that the Viola DX37 invention may not have been fully enabled at the time of publication:

```
            <TITLE>Viola, A Technical Summary</TITLE>
       <CAUTION>THIS DOCUMENT IS IN DRAFT STATUS</CAUTION>
```

For at least the aforementioned reasons, the DX37 Viola files, when considered as a prior art publication for purposes of reexamination, do not teach nor fairly suggest the instant '906 invention, as claimed.

An appendix is attached that presents some of the more relevant Viola documentation files. The files were created for display by a Viola browser and are presented with the included hypermedia tags as found on the CD artifact disk.

## Conclusion

In summary, the Examiner concurs with the Patent Owner with respect to arguments I-IV for the reasons discussed *supra*.

Although the Examiner does not concur with the Patent Owner with respect to argument V, the issue of establishing a nexus between the claimed invention and commercial success is not dispositive.

The Patent Owner's arguments traversing the rejection need only prevail by the "preponderance of the evidence" standard to succeed in having the rejections set forth in the last office action withdrawn. The ultimate determination of patentability must be based on consideration of the entire record, by a preponderance of evidence, with due consideration to the persuasiveness of any arguments and any secondary evidence. In re Oetiker, 977 F.2d 1443, 24 USPQ2d 1443 (Fed. Cir. 1992).

Accordingly, for at least the aforementioned reasons set forth with respect to arguments I-IV, the Examiner has reconsidered and withdrawn the rejections set forth in the last office action (mailed Aug. 16, 2004).

In addition, the DX37 Viola files have been considered as a prior art publication. For the reasons discussed *supra*, the DX37 Viola files included on the CD artifact do not teach nor fairly suggest the instant '906 invention, as claimed.

Instant U.S. Patent 5,838,906 claims 1-10 are hereby confirmed.

Any comments considered necessary by PATENT OWNER regarding the above statement must be submitted promptly to avoid processing delays. Such submission by the patent owner should be labeled: "Comments on Statement of Reasons for Patentability and/or Confirmation" and will be placed in the reexamination file.

St. John Courtenay III
Primary Examiner
Central Reexamination Art Unit 3992

ST. JOHN COURTENAY III
PRIMARY EXAMINER

[illegible]

[illegible] Mark Reinhart, First Conferee
Special Programs Examiner (SPRE)
Central Reexamination Art Unit 3992

Second Conferee
[illegible]

## VIOLA APPENDIX

The contents of file "viola.desc" (contained in the "docs" directory):

```
language:       viola (Visual Interactive O Language and Application)
package:        viola
version:        2.0 beta
parts:          interpreter, applications, documentation,
how to get:     ftp xcf.berkeley.edu src/local/viola/*
description:    A language/toolkit for hypermedia applications. Very loosely
                modeled after Hypercard. Intended as a tool for building
                and running hypermedia applications that are composed of
                collection of classed objects interacting with the user
                and passing messages among each other. Has simple GUI
                specification language. Is event driven (X-Window, timer,
                I/O). Notion of "objects" for modularization and scalability.
                Syntax is C like. Bytecode compilation is done incremental
                (object by object). Is single class inheritanced.
                + has objects
                + dynamic array
                + message passing
                + bytecode compiler, interpreter
                + graphical interface toolkit
                + pseudo-terminal I/O interface
                + socket I/O interface
                + world wide web interface
                - non dynamic class definition (C level definition)
                - non dynamic data types: string, char, int, float, array
                - little interactive authoring tools for naive users
                - development on language is slow, but application driven
ports:          Unix/X
author:         Pei Y. Wei <wei@xcf.berkeley.edu>
status:         actively developed, application driven
discussion:     viola@xcf.berkeley.edu
updated:

-----------
reference:      The X Resources, O'Reilly & Associates

    europe:     ftp info.cern.ch pub/www/src/viola*
    japan:      ftp srawgw.sra.co.jp pub/x11/viola*
```

The file "violaBrief.hmml" (contained in the "docs" directory) provides a technical overview of Viola, and is reproduced for the record in its entirety below (including the "hmml tags" associated with the browser hypermedia markup language):

"violaBrief.hmml" Technical Overview of Viola (see the "docs" directory)

```
<!DOCTYPE hmml SYSTEM>
<HMML>
<TITLE>Viola, A Technical Summary</TITLE>
<CAUTION>THIS DOCUMENT IS IN DRAFT STATUS</CAUTION>
<HSEP>gold</HSEP>
<H1>VIOLA</H1>
<H3>Visual Interactive Object-oriented Langage and Applications</H3>
<P>
This paper presents a technical overview of viola.

<HSEP>gold</HSEP>
<H1>Overview</H1>
<P>
Viola is a tool for the development and support of interactive media
applications. Its basic functionality is not unlike that of
HyperCard and Tcl/Tk. Viola uses an object oriented model for
encapsulating data into ``object'' units, and to enforce a
classing and inheritance system. The extension language is
C-like in syntax, and is compiled into byte-code for
efficient interpretation. The graphical elements (widgets)
exist as classes in the Viola class hierarchy. The set of
widgets implemented in Viola are similar to those found in
graphical user interface toolkits like Xt, plus more
unusual widgets such as HyperCard-like cards and invisible
celopane buttons, and hypertext textfield.

<H2>Classes and Objects</H2>
<P>
The single inheritance classing system defines the basic types of
object instances. Many of these predefined class types happen to be
GUI oriented, because of the current application emphasis on hypermedia,
but many are non-visual and have nothing to do with GUIs. A modular object
model is enforced to control complexity: to provide a relatively simple
way of data encapsualization; for improving the size scalability of viola
applications; and for possibly helping network distribution of objects.
A scripting language exists for application writers to program
modifications to default object behaviors, and for application programming.
<P>
This is the Viola class hierarchy as of this writing.
It is rapidly evolving:
<VOBJF>../apps/chier.v</VOBJF>
```

```
<P>
This class hierarchy seems deficient (at this point and from
some point of view, it's probably true) compared to the GUIs
provided by toolkits like Motif. But, it's actually not as
deficient as it seems. For the same reason as Tk, Viola does
not require hard coding of, for example, dialog boxes to
achieve the same functionality.
<P>
Because of the interpretive nature of the system, complex
GUIs can be composed out of primitive elements, dynamically.
To build a dialog box, a script could be written to create
and necessary objects, and somehow combine them together to
constitute a dialog box.
<P>
Making a dialog box can be made easy by calling a pre written procedure.
The current way to do this in Viola is to build a ``dialog box maker
object'', and to send to it a message:
``Please make me a dialog box, with the following specifications''.

<H3>Hello, World!</H3>
<P>
Here's the proverbial <ITALIC>Hello World</ITALIC> program.
Go ahead, click on it.
<VOBJF>../apps/violaBriefExample_hello.v</VOBJF>
<P>And its file level representation:
<EXAMPLE>
\class {txtButton}
\name {violaBriefExample_hello}
\label {Hello, world!}
\script {
        switch (arg[0]) {
        case "buttonRelease":
                bell();
        break;
        }
        usual();
}
\width {100}
\height {30}
\BGColor {grey45}
\BDColor {white}
\FGColor {white}
\
</EXAMPLE>
<P>
In reality the <CMD>switch()</CMD> would be busier than just handling
one message. But it could just as easily be written thusly (and with
non-essential color information left out):
<EXAMPLE>
\class {txtButton}
\name {hello}
\label {Hello, world!}
```

```
\script {
        if (arg[0] == "buttonRelease") bell();
        usual();
}
\width {100}
\height {30}
</EXAMPLE>
<P>
Although it may seem that some simple binding mechanism would be less
verbose, this free form allows one to easily compose the message
handler in any order -- doing the default action first, then do
the special thing, or any which way.

<H2>Messaging system</H2>
<P>
Viola is message driven, and messages may be generated by a
number of sources. A message is typically caused by the
user interacting with a graphical user interface object,
but it could also be generated by other objects, or by
a timer facility. Through a communication facility such
as the socket, a message may also be generated from another
process on the network.
<P>
In the above ``Hello, world!'' example, when the button is clicked on,
that button object "hello" will eventually receive a "buttonRelease"
message, which according to the script will execute the <CMD>bell()</CMD>
command. If the object does not have any message handlers, the message
will ``fall thru'' the object, and (by way of <CMD>usual()</CMD>)
the class default action will occur.
<P>
A typical viola application consists of a collection of objects interacting
-- generating, receiving, and delegating messages -- with each other, and
with the user.

<H2>The Extension Language</H2>
<P>
As seen in the example above, viola scripts are C-like in syntax.
The language supports way few constructs: <CMD>if, while, for, switch</CMD>.
The commands like <CMD>print(), exit(), create()</CMD>, etc are all
implemented as <ITALIC>methods</ITALIC>. Instead of building the commonly
used commands into the language grammar, they actually are just defined
early enough in the class hierarchy as to be accessible by all subclasses
that may need them.
<P>
All objects can be individually programmed using the scripting
language. Each object is essentially its own interpretive
environment, and each object is its own variable scope.
<P>
For optimization, object scripts are compiled into <ITALIC>byte
codes</ITALIC> before applying the byte code interpreter on them. Because an
object's script is basically a message event handler that is likely to
receives many messages in its instance life time, the one time
```

```
cost of parsing and simple transformation into byte codes is
very worthwhile. The gain in execution speed is especially
apparent when the objects deal with time critical "mouseMove"
messages, or if there are tight looping operations.

<HSEP>gold</HSEP>
<H1>Applications</H1>
<P>
Along side the development of the Viola language/toolkit
<ITALIC>engine</ITALIC> itself, there is also the development of real
working applications using the engine. The two processes provide reality
checks for each other.
<P>
Here we show screendumps of two developing viola applications.

<H2>World Wide Web Browser</H2>
<FIGURE TYPE="image/gif" SRC="../docs/violaWWW.gif">
<P>
This ``ViolaWWW'' application is currently among the most actively developed
viola application. The initial viola-WWW effort was made in order to
provide to viola a clean network transport mechanism. But the ViolaWWW
browser application itself turned out to be useful enough, that it is being
actively developed, with emphasis on support for online publishing.
<P>
An early version of this browser has been in use in the WWW community
since mid 92, it being the first publically available World Wide Web
browser for X-Windows.

<H2>The Whole Internet Resource Catalog</H2>
<FIGURE TYPE="image/gif" SRC="twi.gif">
<P>
An electronic version of the resource catalog portion of the book
<ITALIC>The Whole Internet</ITALIC>. This application uses HyperCard
style card-flipping technique to flip among four basic GUI sets
(the cover frame is shown here; others frames contain documents and
controlling GUI elements).

<HSEP>gold</HSEP>
<H1>Summary</H1>
<P>
In sum, the Viola language/toolkit system provides an environment
where applications are composed of groups of objects, where objects
interact, by message passing, with the user and with each other.
<P>
As more applications are developed, more reusable objects will be created.
And development of successive applications will become easier and easier.
One of the goals of the Viola project is to accumulate a collection of
objects useful for constructing hypermedia applications.
<P>
The immediate future direction of Viola development will continue to aim
towards the path of hypermedia applications, with the World Wide Web
as the document/object network transport infrastructure.
```

```
<P>
If you're interested in contributing to the development effort,
please contact me.
<HSEP>gold</HSEP>
<ADDRESS>
<P>Pei Y. Wei
<P>Developer, O'Reilly & Associates, Digital Media Group
<P><CMD>wei@ora.com</CMD>
</ADDRESS>
</HMML>
```

The contents of file "violaCh1.hmml" (contained in the "docs" directory):

```
<!DOCTYPE hmml SYSTEM
[
]>
<HMML>
<SECTION NAME="chapter1">
<H1>Introduction to Viola</H1>
<FIGURE TYPE="image/xbm" SRC="viola.xbm">

<SECTION NAME="whatIsViola">
<H2>What is Viola?</H2>
<P>
Viola is a hypermedia application authoring and supporting system.
It contains a graphical user interface set, an ``object oriented''
data organization and storage model, and a built-in extension
language. Perhaps the most important contribution of Viola is its
potential in bringing HyperCard-like capability to a very wide
range of platforms.
<p>
Viola can be used for the development and support of
interactive media applications. It provides an object data
organization model, an interpreted extension language,
graphical elements for user interface. The Viola operating
environment is interpretive, designed for rapid prototyping.
<P>
Viola is desigend to aid the development and support of
interactive/hyper media applications for the Unix/X platform.
Its functionality is similar to HyperCard and Tcl/Tk.
Viola uses an object oriented model to facilitate data
encapsulation into ``object'' units, and to enforce a
 classing and inheritance system. The extension language is
C-like in syntax and is processed into byte-code for
efficient interpretation. The graphical elements (widgets)
exist as classes in the Viola class hierarchy. The set of
widgets implemented in Viola are similar to those found in
user interface toolkits like Xt, plus more unusual widgets
such as HyperCard-like cards and invisible celopane buttons,
and hypertext textfield.
```

```
<P>
In sum, Viola provides an environment in which applications
are composed of groups of objects where each object interacts,
by message passing, with the user and with each other.
<P>
Because most aspects of an object is accessible and controllable
through the interpreted extension language, building an
application in Viola can be done dynamically, without the
edit/compile cycle. As with other systems with built-in
extension language (Emacs/ELisp, Tk/Tcl, HyperCard/HyperTalk),
Viola derives much of its versitility from its extension
language.
<P>
The rest of this paper gives a brief overview of the Viola
basics: the object model, language, and GUI elements.
It also describes some applications(?).
</SECTION>

<SECTION NAME="objectSystem">
<H2>The Object System</H2>
<P>
This section briefly describes Viola's notion of object
orientation.
<P>
Each Viola object consists of an array of ``slot'' values.
These values are information pertaining specifically to the
object: its class, name, script, color, and so on. The number
and type of each slot in an object are determined by the class
of the object.
<P>
Each class inherits slot definitions from its superclasses,
and has the option to set new values for the inherited slots.
In addition to those inherited slots, it may define two types
of new slots: private and common.
<P>
Common slots define slots that are shared by all object
instances of the same class. Private slots define slots that
make up each object instance. The separation of common and
private slots reduces redundancy of information carried by
each object.
<P>
As with slots, class methods are also inherited. The idea,
again, is to provide a mechanism for sharing as much code
as possible. It also makes the task of subclasing relatively
easy and systematic. It should be noted that modification of
the object system (to subclass, adding slots and methods)
must, at this point, be done in C.
<P>
This is the Viola class hierarchy as of this writing. It is
evolving rapidly.
</SECTION>
```

```
<SECTION NAME="violaClassHierarchy">
<H2>The Viola Class Hierarchy</H2>
<EXAMPLE>
        cosmic
                generic
                        field
                                BCard
                                FCard
                                XBM
                                        XBMButton
                                        toggle
                                XPM
                                        XPMButton
                                GIF
                                dial
                                client
                                        TTY
                                        socket
                                menu
                                pane
                                        hpane
                                                txt
                                                txtLabel
                                                txtButton
                                                txtDisp
                                                txtEdit
                                        vpane
                                project
                                rubber
                                slider
                                stack
                                tray
</EXAMPLE>

<P>
The cosmic class defines the minimal object: a private slot
that lets the object know what class it belongs to; and essential
methods such as create(), destroy(), save(), etc. From here on
the slots and methods definition is rather arbitrary and depends
on what the application is.
<P>
As Viola was designed for visually interactive applications,
most of the classes are GUI widgety oriented. The two notable
exceptions are the socket and TTY classes, which are useful
for communicating with other processes.
<P>
The class hierarchy seems deficient (at this point and from
some point of view, it's probably true) compared to the GUIs
provided by toolkits like Motif. But, it's actually not as
deficient as it seems. For the same reason as Tk, Viola does
not require hard coding of, for example, dialog boxes to
achieve the same functionality.
```

```
<P>
Because of the interpretive nature of the system, complex
GUIs can be composed out of primitive elements, dynamically.
To build a dialog box, a script could be written to create
and necessary objects, and somehow combine them together to
constitute a dialog box.
<P>
As in Tk, making a dialog box can be made easy by calling a
pre written procedure. The current way to do this in Viola is
to build a ``dialog box maker object'', and to send to it a
``Please make me a dialog box, with the following specifications''.
<P>
It's worthwhile to illustrate with an example, which will
show many other aspects of Viola.

</SECTION>

<SECTION NAME="hello.v">
<H2>hello.v</H2>
<EXAMPLE>
\class {txtButton}
\name {hello}
\label {Hello, world!}
\script {
        switch (arg[0]) {
        case "buttonRelease":
                res.dialog("show",
                           "Are you sure you want to exit?",
                           "Yes",       "callback_exit",
                           "No",        "callback_nevermind");
        break;
        case "callback_exit":
                exit(0);
        case "callback_nevermind":
                return; /* do nothing */
        }
        usual();
}
</EXAMPLE>

</SECTION>
</SECTION>

</HMML>
```

**How to Contact the Examiner:**

Any inquiry concerning this communication or earlier communications from the examiner should be directed to St. John Courtenay III, whose telephone number is 571-272-3761. A voice mail service is also available at this number. The Examiner can normally be reached on Monday - Friday, 9:00 AM - 5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the Examiner's Supervisor, Mark Reinhart who can be reached at 571-272-1611.

The fax phone number for the organization where this application or proceeding is assigned is:

**CENTRAL REEXAMINATION UNIT FAX NUMBER:**

**571-273-9900**

**All responses sent by U.S. Mail should be mailed to:**

Commissioner for Patents
PO Box 1450
Mail Stop ex parte REEXAM
Alexandria, VA 22313-1450

[illegible]

ST. JOHN COURTENAY III
PRIMARY EXAMINER

# 831 PH Ex. 20

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 90/006,831 | 10/30/2003 | 5838906 | | 9718 |

30080 7590 01/20/2006

LAW OFFICE OF CHARLES E. KRUEGER
P.O. BOX 5607
WALNUT CREEK, CA 94596-1607

| EXAMINER |
|---|
| |

| ART UNIT | PAPER NUMBER |
|---|---|
| | |

DATE MAILED: 01/20/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

| | | |
|---|---|---|
| ***Notice of Intent to Issue*** SUPPLEMENTAL ***Ex Parte Reexamination Certificate*** | **Control No.** 90/006,831 | **Patent Under Reexamination** 5838906 |
| | **Examiner** St. John Courtenay III | **Art Unit** 3992 |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

1. ☒ Prosecution on the merits is (or remains) closed in this *ex parte* reexamination proceeding. This proceeding is subject to reopening at the initiative of the Office or upon petition. *Cf.* 37 CFR 1.313(a). A Certificate will be issued in view of
   (a) ☒ Patent owner's communication(s) filed: *12 October 2004*.
   (b) ☐ Patent owner's late response filed: _____.
   (c) ☐ Patent owner's failure to file an appropriate response to the Office action mailed: _____.
   (d) ☐ Patent owner's failure to timely file an Appeal Brief (37 CFR 41.31).
   (e) ☐ Other: _____.

   Status of *Ex Parte* Reexamination:
   (f) Change in the Specification: ☐ Yes ☐ No
   (g) Change in the Drawing(s): ☐ Yes ☐ No
   (h) Status of the Claim(s):
   (1) Patent claim(s) confirmed: *1-10*.
   (2) Patent claim(s) amended (including dependent on amended claim(s)): _____
   (3) Patent claim(s) cancelled: _____.
   (4) Newly presented claim(s) patentable: _____.
   (5) Newly presented cancelled claims: _____.

2. ☒ Note the attached statement of reasons for patentability and/or confirmation. Any comments considered necessary by patent owner regarding reasons for patentability and/or confirmation must be submitted promptly to avoid processing delays. Such submission(s) should be labeled: "Comments On Statement of Reasons for Patentability and/or Confirmation."

3. ☒ Note attached NOTICE OF REFERENCES CITED (PTO-892).

4. ☐ Note attached LIST OF REFERENCES CITED (PTO-1449 or PTO/SB/08).

5. ☐ The drawing correction request filed on _____ is: ☐ approved ☐ disapproved.

6. ☐ Acknowledgment is made of the priority claim under 35 U.S.C. § 119(a)-(d) or (f).
   a)☐ All b)☐ Some* c)☐ None of the certified copies have
   ☐ been received.
   ☐ not been received.
   ☐ been filed in Application No. _____.
   ☐ been filed in reexamination Control No. _____.
   ☐ been received by the International Bureau in PCT Application No. _____.

   * Certified copies not received: _____.

7. ☐ Note attached Examiner's Amendment.

8. ☒ Note attached Interview Summary (PTO-474).

9. ☐ Other: _____.

St. John Courtenay III
ST. JOHN COURTENAY III
PRIMARY EXAMINER

St. John Courtenay III
Primary Examiner
Art Unit: 3992

cc: Requester (if third party requester)

UNITED STATES DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE

# REEXAMINATION

## REASONS FOR PATENTABILITY / CONFIRMATION

Reexamination Control No. 90/006,831 Attachment to Paper No. 20050823.

Art Unit 3992.

See attached "Examiner's Statement of Reasons for Patentability / Confirmation."

ST. JOHN COURTENAY III
PRIMARY EXAMINER

[illegible]

(Examiner's Signature)

PTOL-476 (Rev. 03-98)

**Examiner's Statement of Reasons for Patentability and/or Confirmation**

The following is an Examiner's statement of reasons for patentability and/or confirmation of the claims found patentable in this reexamination proceeding.

**Summary**

At the outset, it is noted that the previous Examiner of record admitted in making the rejection under 35 U.S.C. §103 of independent claims 1 and 6 that the cited four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II, does not explicitly teach a method that enables interactive processing of an object:

> The combination of patentee'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, with distributed hypermedia documents [see Office Action mailed Oct. 16, 2004, page 6, lines 18-21].

The previous Examiner then applied a fifth reference (Toye) to the combination and asserted:

> 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 (See Toye: p. 40 description of NoteMail, particularly p. 40, col. 2, first paragraph).

An Examiner's statement of reasons for confirmation and/or patentability is set forth below in the form of a reply to the Patent Owner's detailed arguments of record. The Patent Owner's arguments are shown in *italics* below. In addition, the "DX37" Viola code has been considered by the PTO as a prior art publication. The Viola code issue is addressed at the end of the response to the Patent Owner's detailed argument.

## Examiner's Response to Patent Owner's Detailed Argument

*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.*

In response, the Examiner finds the Patent Owner's argument I(a) persuasive for at least the following reasons:

As acknowledged by the previous Examiner, the cited four-way combination of the patent owner's admitted prior art (APA), 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, with distributed hypermedia documents" [see Office Action mailed Oct. 16, 2004, page 6, lines 18-21].

During patent examination, the pending claims must be "given their broadest reasonable interpretation consistent with the specification." In re Hyatt, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000).

Accordingly, with respect to the scope of the claimed "interactive processing," the Examiner must apply the broadest reasonable interpretation consistent with the specification.

To be consistent with the specification, the claimed "interactive processing" necessarily requires some capability of ongoing real-time manipulation and control by the user of the object displayed within the browser-controlled window.

In particular, the claimed "interactive processing," when properly construed in a manner consistent with the specification, requires:

> "**Interprocess communication** between the **hypermedia browser** and the **embedded application program** is **ongoing** after the program object has been launched" [see instant '906 patent, col. 7, lines 1-4].

Static objects disclosed by the prior art of record, such as graphical images of mathematical formulas (see e.g., the use of the EMBED tag in Raggett I at the bottom of page 6) are incapable of providing "interactive processing" as required by the instant '906 claims because the application that renders the static object terminates after the rendering step and prior to the complete display of the web page.

With respect to prior art of record that uses colored or otherwise identifiable active areas superimposed on a coordinate grid of a static image map (e.g., see the use of the "ismap" attribute and "<figt " tag in Raggett I - see "Active areas" on page 13; see also U.S. Patent 4,847,604 to Doyle), these "map" images are created by an executable rendering application that

generates the static "map" image and then terminates prior to the complete display of the web page.

The aforementioned prior art "map" images are static in the sense that the user cannot interactively change the appearance of the "map" image, but are also active in the sense that the user can interactively click on an active region or area within the map and trigger a URL that is invoked by the web browser application.

Significantly, with respect to active maps and the like, it is the browser application (i.e., not an executable application separate from the browser application) that makes the active areas "interactive" by waiting for user input, typically in the form of a mouse click [see e.g., Raggett I, page 13, 1st sentence under "Active areas"].

Because the aforementioned prior art executable rendering applications terminate after generating the static image, it is axiomatic that there is no ongoing interprocess communication between the browser and the executable application. Therefore, there is no ongoing real-time manipulation and control by the user of the object displayed within the browser-controlled window, as required by the instant '906 claims when the claim element "interactive processing" is properly construed in a manner consistent with the specification of the '906 patent.

The instant '906 patent specification makes liberal use of the term "interactive" as being synonymous with "manipulate" and "control" in an ongoing real-time setting:

See '906 Patent, col. 6, lines 40-47:

> Thus, it is desirable to have a system that allows a user at a small client computer connected to the Internet to locate, retrieve and **manipulate** data objects when the data objects are bandwidth-intensive and compute-intensive. Further, it is desirable to allow a user to **manipulate** data objects in an **interactive** way to provide the user with a better understanding of information presented and to allow the user to accomplish a wider variety of tasks.

See '906 Patent, col. 6, lines 50-62:

> The present invention provides a method for running embedded program objects in a computer network environment. The method includes the steps of providing at least one client workstation and one network server coupled to the network environment where the network environment is a distributed hypermedia environment; displaying, on the client workstation, a portion of a hypermedia document received over the network from the server, where the hypermedia document includes an **embedded controllable application**; and **interactively controlling** the **embedded controllable application** from the client workstation via communication sent over the distributed hypermedia environment.

See '906 Patent, col. 6, lines 63-67 cont'd col. 7, lines 1-6:

> The present invention allows a user at a client computer connected to a network to locate, retrieve and **manipulate objects** in an **interactive way**. The invention not only allows the user to use a hypermedia format to locate and retrieve program objects, but also allows the user to **interact** with an application program located at a remote computer. **Interprocess communication between the hypermedia browser and the embedded application program is ONGOING after the program object has been launched**. The user is able to use a vast amount of computing power beyond that which is contained in the user's client computer.

See '906 Patent, col. 9, line 66 cont'd col. 10, lines 1-16:

> After application client 210 receives the multidimensional data object 216, application client 210 executes instructions to display the multidimensional embryo data on the display screen to a user of the client computer 200. The user is then able to **interactively operate controls to recompute different views for the image data**. In a preferred embodiment, a control window is displayed within, or adjacent to, a window generated by browser client 208 that contains a display of hypermedia document 212. An example of such display is discussed below in connection with FIG. 9. Thus, **the user is able to interactively manipulate a multidimensional image object** by means of the present invention. In order to make application client 210 integral with displays created by browser client 208, both the browser client and

> the application client must be in communication with each other, as shown by the arrow connecting the two within client computer 200. The manner of communication is through an application program interface (API), discussed below.

See '906 Patent, col. 10, lines 47-56:

> In the present example where a multidimensional image object representing medical data for an embryo is being viewed, application server 220 could perform much of the viewing transformation and volume rendering calculations to **allow a user to interactively view** the embryo data at their client computer display screen. In a preferred embodiment, application client 210 **receives signals from a user input device** at the user's client computer 200. **An example of such input would be to rotate the embryo image from a current position to a new position from the user's point of view**.

See '906 Patent, col. 16, lines 18-20.

> FIG. 9 is a screen display of the invention showing an **interactive application object (in this case a three dimensional image object)** in a window within a browser window. In FIG. 9, the browser is NCSA Mosaic version 2.4. The processes VIS, Panel and VRServer work as discussed above. FIG. 9 shows screen display 356 Mosaic window 350 containing image window 352 and a portion of a panel window 354. Note that <u>image window 352 is within Mosaic window 350</u> while panel window 354 is external to Mosaic window 350. Another possibility is to have panel window 354 within Mosaic window 350. By **using the controls** in panel window 354 **the user is able to manipulate the image within image window 352 in REAL TIME to perform such operations as scaling, rotation, translation, color map selection, etc.**

The Examiner submits that this interpretation is reasonable and also consistent with the interpretation that those skilled in the art would reach. See <u>In re Cortright</u>, 165 F.3d 1353, 1359, 49 USPQ2d 1464, 1468 (Fed. Cir. 1999), "The broadest reasonable interpretation of the claims must also be consistent with the interpretation that those skilled in the art would reach."

The above discussion does not mean that the use of static objects precludes interactivity. One may reasonably argue that the use of static graphical images that contain superimposed active areas or sections (e.g., through the use of the "ismap" attribute and "<figt " tag in Raggett I, *supra*) enable "interactive processing" in the sense that when a user clicks the mouse over an active area, a URL call to a server is generated by the browser; however, <u>this is not the same kind of "interactive processing" required by instant claims 1 and 6 of the '906 patent</u>.

In the case of the Raggett I "*ismap*" attribute, Raggett explicitly discloses:

> "The *ismap* attribute causes <u>the browser to send mouse clicks on the figure, back to the server</u> using the selected coordinate scheme" [see Raggett I, page 13, 1st sentence under "Active areas"].

As is clearly indicated by Raggett I, it is the <u>browser application</u> that responds to the mouse click that occurs over an active region identified by a coordinate scheme superimposed over a static graphical image. Thus, in the case of Raggett I and active map areas in general (e.g., using the "ismap" attribute and "<figt " tag), it is the <u>browser application</u> that provides the interactivity.

In contrast, the instant '906 claims explicitly require the "interactive processing" to be enabled by an "executable application" that is a <u>separate application</u> from the browser application.

The instant claimed '906 "executable application" that provides the claimed "interactive processing" is invoked <u>not in response to a user event detected by the browser</u> (as in the case of Raggett I, *supra)*, but rather in response to the browser application parsing an "embed text format" (i.e., an "EMBED" tag, see col. 12, line 60, '906 patent) that is detected within the hypermedia document when the hypermedia document is first loaded by the browser.

Significantly, the instant claimed "interactive processing" of the '906 patent begins at the moment the browser application parses an "embed text format" detected within the hypermedia document. The web browser invokes the claimed "executable application" immediately after an "EMBED" tag is parsed and before the hypermedia document is completely displayed in the browser-controlled window. The invoked "executable application" enables the claimed "interactive processing."

Instant '906 independent claims 1 and 6 therefore require an <u>operative coupling</u> between the claimed "executable application" and the claimed "interactive processing" such that the claimed "interactive processing" must be enabled by an "executable application" that meets five explicitly claimed requirements:

1. The executable application must be external to the first distributed multimedia document.
2. The executable application must be automatically invoked by the browser application when the "embed text format" is <u>parsed</u> by the browser application.
3. The executable application must execute on the client workstation.
4. The executable application must display the object within the display area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser-controlled window.

5. The executable application must enable interactive processing of the object within the display area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser-controlled window.

Because the admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II four-way combination displays or renders a static image and then terminates, "interactive processing" as used in the instant claims is precluded by the four-way combination.

As discussed *supra,* a proper construction of the claimed "interactive processing" necessarily requires some capability of ongoing real-time manipulation and control by the user that is applied to the object displayed within the first browser-controlled window. It is axiomatic that an executable application that terminates is incapable of providing the type of "interactive processing" required by instant '906 independent claims 1 and 6.

In particular, executable application requirement #5, *supra,* is clearly not met by the cited four-way combination of admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II, with respect to the operative coupling required between the claimed "executable application" and claimed "interactive processing."

**THE TOYE REFERENCE**

The Examiner finds the Patent Owner's argument (as supported by the Felten II affidavit, §§33-35) persuasive that Toye teaches the use of an image or icon that represents a file or data object displayed within a "NoteMail" page, and that the image or icon consists of a "static snapshot" of the external content. Interactive processing is enabled only after a user manually clicks on the "static snapshot" image to launch an external editor program.

Toye discloses <u>manual selection</u> by the user to enable interactivity:

> "Subsequently <u>selecting</u> the displayed data <u>with a mouse</u> will **restart** the original application, so that the data can be edited or updated without leaving the notebook environment" [See Toye, p. 40, 2nd column, 2nd paragraph].

Significantly, Toye discloses functionality similar to a file manager:

> "The functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file" [p. 40, 2nd column, 2nd paragraph].

The Examiner concurs with the Patent Owner's contention that no ongoing interaction with the data can occur unless the "appropriate application" is manually <u>started</u> or <u>restarted</u> by the user to enable interaction with the data displayed as a static "snapshot image" or icon within a "NoteMail" page.

The Examiner concurs that automatic invoking, as taught by Toye, is the result of manual user selection with a mouse of a "static snapshot" image that automatically launches the "appropriate application" to edit the data object. This approach appears to be similar to the method employed by conventional file manager programs that implement file type association to invoke the appropriate application when the user clicks on the filename or file icon.

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

> *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 response, the Examiner finds the Patent Owner's argument I(b) persuasive for at least the following reasons:

The Examiner concurs with the Patent Owner's argument regarding the Raggett I & II EMBED tag that is located at a first location in a hypermedia document. When the EMBED tag is parsed, a rendering application is invoked that returns a STATIC graphical image to be displayed within the browser window at the first location, and then the rendering application terminates prior to the complete display of the web page.

Because the application terminates after rendering the graphical object, it is clear that a terminated rendering application is incapable of providing the claimed "interactive processing," as discussed *supra*.

With respect to the cited Toye reference, the Examiner has considered Professor Felton's affidavit ("Felton II" at paragraph 38) supporting the Patent Owner's contention that Toye teaches away from the proposed combination because existing editor applications at the time of the '906 invention were designed to run in their own dedicated windows.

Whether Toye teaches away with respect to the superimposed display of the X-server output within the "NoteMail" viewer is a close question [see Toye, p. 40, 2nd paragraph, i.e., "any application that displays through an X-server can insert its output (audio, video, or graphics) dynamically onto a notebook page through an embedded 'virtual window.' " ]. The question turns upon whether the Toye "NoteMail" viewing system is equivalent to the browser claimed in the '906 patent and also whether the "embedded virtual window" disclosed by Toye is equivalent to displaying an object within a display area of the "browser-controlled window" claimed in the '906 patent [claims 1 & 6].

As disclosed by Toye, the "NoteMail" system is a hybrid tool that combines "the functions of an engineering notebook, hypermedia browser and authoring environment, mail tool, and file application manager" [Toye, p. 40, col. 1, 3rd paragraph]. With respect to the first prong (i.e., whether the Toye "NoteMail" viewing system is equivalent to the browser claimed in the '906 patent) reasonable arguments may be proffered on both sides.

One might reasonably conclude that the Toye "NoteMail" system is a specialized hypermedia browser, i.e., a species of the genus of hypermedia browsers. "A generic claim cannot be allowed to an applicant if the prior art discloses a species falling within the claimed genus." The species in that case will anticipate the genus. In re Slayter, 276 F.2d 408, 411, 125 USPQ 345, 347 (CCPA 1960); In re Gosteli, 872 F.2d 1008, 10 USPQ2d 1614 (Fed. Cir. 1989).

On the other hand, if the engineering notebook, authoring environment, mail tool, and file application manager functions are the dominant functions of the Toye "NoteMail" viewer, then one could reasonably argue that Toye does not teach the hypermedia browser required by the instant '906 claims when the claims are properly interpreted by applying the broadest reasonable interpretation consistent with the specification. However, the issue of how the instant '906 "hypermedia browser" is construed is not dispositive.

The second prong of inquiry is also a close question, i.e., whether the "embedded virtual window" disclosed by Toye is equivalent to displaying an object at a first location within the display area of the "browser-controlled window" as claimed in the '906 patent. Toye provides further insight regarding the implementation of the "embedded virtual window" by explicitly citing the MediaMosaic article [see Toye, p. 40, col. 2, 3rd paragraph, i.e., "We are aware of only one other multimedia editor with such an architecture, MediaMosaic (22)"].

While Professor Felton's affidavit is technically correct in asserting that existing editor applications at the time of the '906 invention were designed to run in their own dedicated windows, the "virtual window" system disclosed in the MediaMosaic article provides further implementation details. Accordingly, MediaMosaic has been considered by the Examiner as extrinsic evidence to aid in the interpretation of the cited Toye reference. [1]

Felton II at 38 argues:

> 38. Indeed, Toye teaches the use of external editor programs that have not been modified from their standard versions. (See, e.g., Toye at p. 40, col. 2, first full paragraph: "any application that displays through an X-server") Such unmodified programs are not suitable for use within an enclosing document display, because the unmodified programs conventionally display menus and button bars at the top, and other graphical elements around their edges. External application windows with these elements on their borders cannot naturally be displayed within a document display; at most they could be displayed in a window area elsewhere in a windowing environment, as discussed in the previous paragraph. To enable a reasonable editing experience within a document display, the applications would have to be modified; but Toye teaches that they are not modified.

Page 136 of the MediaMosaic article reveals how embedded virtual screens (i.e., embedded virtual windows) were implemented at the time of the Toye reference. MediaMosaic reveals that a virtual screen is a pseudo root window to map X clients so that a portion of their output screens can be embedded in a document as a general media container." Virtual screens use a "pseudo server" that "intercepts and modifies X protocols between X clients and the X server." The protocol essentially "reparents clients to a designated window

[1] Lin, J.K., "MediaMosaic – A Multimedia Editing Environment", Proc. 5th Annual Symposium on User Interface Software and Technology, Monterey CA, Nov. 15-18, 1992 (published by ACM Press).

instead of the root window of the real screen." MediaMosaic creates a virtual screen for a media client and embeds it in a document. MediaMosaic further creates a user-movable and resizable "Viewport" (X Window) for each embedded virtual screen.

The embedded virtual screen is mapped to its corresponding "Viewport" before it is inserted into a document. Text in the document is automatically reformatted around the inserted media displayed within the "Viewport." Significantly, "The mechanism used by MediaMosaic to contain general media is to directly embed them in documents <u>by their original displaying tools</u>" [MediaMosaic, p. 138, 1st paragraph under "5 Duplicated and Full Views].

It is reasonable to assume that Toye uses the MediaMosiac "virtual screen" embedding method because Toye explicitly states that MediaMosaic has the same architecture (i.e., as "NoteMail") [see Toye, p. 40, col. 2, §2].

Prof. Felton's assertion that the applications would have to be modified "because the unmodified programs conventionally display menus and button bars at the top, and other graphical elements around their edges" [see Felton II at 38] is contravened by the extrinsic evidence that MediaMosaic uses the original unmodified rendering tools (i.e., the associated editing applications) to directly embed output media in documents. MediaMosiac simply redirects a portion of the application display output (containing the object to be embedded) to a "virtual screen" associated with a mapped "Viewport."

MediaMosiac appears to operate by cropping out the portion of the application display output that contains the aforementioned display menus, button bars, and other graphical elements around the edges that are normally displayed in the full screen mode of an editing program. Only the embedded object of interest is displayed within the virtual screen associated with the mapped "Viewport" and no modification of the rendering editing application appears to be required [e.g., see Fig. 4, p 139].

MediaMosaic provides an alternate user-selectable full view mode for editing embedded media, as manual resizing of a Viewport window is an awkward way to access the full controls of an associated editing application [see Fig. 5, p. 139].

MediaMosaic therefore provides a mechanism to allow users to embed data objects displayed by different editing applications into one document. Significantly, the system disclosed by MediaMosaic provides the capability to "tailor" (i.e., edit or control) the individual embedded data objects <u>by direct manipulation</u> [MediaMosaic, p. 140, 1st col., §2].

MediaMosaic does enable interactive control and manipulation of objects embedded in what arguably may be construed to be a "browser-controlled window," BUT ONLY AFTER USER INTERVENTION, such as by making a selection with a mouse.

MediaMosaic explicitly discloses: "<u>users</u> can switch media modes <u>by selecting</u> 'Full-View Editing' or 'Embedded-View Editing' from the pull-down menu."

Likewise, Toye teaches that interactive processing is enabled only after a user manually clicks on the "static snapshot" image to launch an external editor program, as discussed *supra.*

Significantly, the prior art approaches of both Toye and MediaMosaic require user intervention to launch an executable application to enable interactive processing. In contrast, the instant '906 claims do not require user intervention to launch the executable application that enables the claimed "interactive processing." Accordingly, for at least this reason, Toye does not anticipate nor render obvious the instant '906 invention.

> *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 6.*

In response, the Examiner finds the Patent Owner's argument I(c) persuasive for at least the following reasons:

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. In re Royka, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." In re Wilson, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970).

The Examiner concurs with the Patent Owner's argument that the proposed five-way combination of references set forth in the last office action does not show automatic invocation of the executable application that enables interactive processing when the hypermedia document is parsed, as claimed.

As persuasively argued by the Patent Owner, the proposed five-way combination of references 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 cited four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II teaches that external data is rendered to a static bit map that is displayed by the browser.

In contrast, Toye teaches that external data is displayed as a "static snapshot" (i.e., representing a data object) within a NoteMail page <u>that must be selected by a mouse to launch an editor application in a separate window</u>" [see Felten II, at paragraph 47]. Thus, Toye clearly requires user intervention to enable interactive processing.

For the aforementioned reasons, the Examiner agrees that all claim limitations are not taught nor fairly suggested by the combination of cited references. Accordingly, the combination proposed in the rejection does not include all the limitations of claims 1 and 6 and a *prima facie* case of obviousness has not been established.

> *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.*

In response, the Examiner finds the Patent Owner's argument II(a) persuasive for at least the following reasons:

As noted *supra,* the previous Examiner of record admitted in making the rejection under 35 U.S.C. §103 of independent claims 1 and 6 that the cited four-way combination of the patent owner's admitted prior art (APA), 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, with distributed hypermedia documents"* [see Office Action mailed Oct. 16, 2004, page 6, lines 18-21].

The Examiner concurs with the Patent Owner's argument that the addition of the Toye reference is a contradiction, and therefore teaches away, from the four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II, because, as the Patent Owner points out, combining Toye with aforementioned four-way combination "would change the principle of operation of the Mosaic, Berners-Lee, Raggett I 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" [see Patent Owner's response, Oct. 12, 2004, page 15, last paragraph].

For at least the aforementioned reason, the cited Toye reference teaches away from the four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II.

> *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.*

In response, the Examiner finds the Patent Owner's argument II(b) persuasive for at least the following reasons:

The Patent Owner points out that "the Mosaic [APA], 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 [see Patent Owner's response, Oct. 12, 2004, page 16, third paragraph from the bottom of the page].

The Patent Owner further observes: "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" [see Patent Owner's response, Oct. 12, 2004, page 16, second from last paragraph].

The Patent Owner further concludes that "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, <u>thereby destroying its distributed nature</u>. 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" [see Patent Owner's response, Oct. 12, 2004, page 17, second from last paragraph].

The Examiner concurs with the Patent Owner that the centralized collaborative access system disclosed by Toye teaches away from the distributed system that allows objects to be stored anywhere on the Internet, as taught by the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II. The Examiner agrees that the centralized database approach of Toye has no applicability to the distributed system of the cited Mosaic (APA), Berners-Lee, Raggett I and II combination, and thus Toye teaches away from the four-way combination. A *prima facie* case of obviousness may be rebutted by showing that the art, in any material respect, teaches away from the claimed invention. In re Geisler, 116 F.3d 1465, 1471, 43 USPQ2d 1362, 1366 (Fed. Cir. 1997).

The Examiner finds that the proposed modification would render the prior art invention being modified (i.e., the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II) unsatisfactory for its intended purpose if combined with Toye. The purpose of the Toye centralized collaborative database (i.e., "a collaborative tool for creating, viewing, and sharing multimedia engineering documents in a network environment", see Toye p. 40, col. 1) is distinctly different than the purpose of the cited four-way combination browser that can access another document located anywhere on the Internet.

In contrast, Toye explicitly discloses: "Applications can now reside anywhere on the Internet" as opposed to accessing documents located anywhere on the Internet, as taught by the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II [see Toye, p. 40, col. 2, 2nd paragraph, last line].

Accordingly, Toye teaches away from the cited four-way combination by rendering it unsatisfactory for its intended purpose. If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification. In re Gordon, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984).

Because the system of Toye relies upon a centralized collaborative database as a fundamental principle of operation, and the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II teaches the use of a distributed system that allows objects to be stored anywhere on the Internet, the proposed modification by Toye of the prior art (i.e., Mosaic (APA), Berners-Lee, Raggett I and II) would clearly change the principle of operation of the prior art invention being modified. If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious. In re Ratti, 270 F.2d 810, 123 USPQ 349 (CCPA 1959).

> *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.*

In response, the Examiner finds the Patent Owner's argument II(c) persuasive for at least the following reasons:

As pointed out by the Patent Owner, the Toye reference teaches a system that is appropriate for a collaborative workgroup where the participants know and trust each other and where all authorized users may access and modify the collaborative document after its creation.

The Examiner concurs with the Patent Owner's argument that the publish-once/view-many paradigm that preserves the data and referential integrity (i.e., unidirectional links) defined by the web document author (i.e., as taught by the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II) is destroyed by the modification suggested by the Toye reference.

The addition of the Toye reference clearly teaches away from the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II, because Toye renders the prior art invention being modified unsatisfactory for its intended purpose of preserving the data and referential integrity (i.e., unidirectional links) defined by the web document author. If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification. In re Gordon, 733 F.2d at 900.

*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.*

As disclosed by Toye, NoteMail "combines the functions of an engineering notebook, hypermedia browser, and authoring environment, mail tool, and file application manager" [see Toye, p. 40, col. 1].

Toye implements a Distributed Information Service (DIS) that Toye defines as follows:

> 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 server as a reference pointer or "handle" [see Toye, page 40, 2nd column, 2nd from last paragraph].

However, the Patent Owner argues:

> 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." ['906 at col. 5, lines 25-38].

The Felton II affidavit further argues:

> Toye does not teach the use of a 'distributed hypermedia environment,' as that term is used in the '906 claims. The environment provided by Toye is not distributed in the sense of the '906 claims, since it relies on the centralization of a user's document storage in one place. Toye teaches away from the use of a distributed hypermedia environment." (see Felton II, paragraph 25).

The above characterization in Felton II (i.e., "Toye teaches away from the use of a distributed hypermedia environment") is somewhat counterintuitive because Toye teaches the use of combined functions that explicitly include the functions of a "hypermedia browser," and Toye also uses the term "Distributed" in labeling the "Distributed Information Service" [see Toye, p. 40, col. 2].

It appears that the moniker "Distributed" may have been used in labeling Toye's "Distributed Information Service" because centralized information and management services may be distributed to users, e.g., via "persistent objects" that are created in DIS to serve as a reference pointers or handles [see Toye, p. 40, col. 2, 2nd from last paragraph].

The Examiner does not agree with the Patent Owner's assertion that "NoteMail" pages are "not analogous" to Web-style hypermedia documents [see p. 21, 4th paragraph].

Toye explicitly discloses that NoteMail "combines the functions of an engineering notebook, <u>hypermedia browser</u>, and authoring environment, mail tool, and file application manager" [see Toye, p. 40, col. 1].

Toye explicitly discloses the use of "hyper-documents" in the context of an "Internet-wide information web":

> Messages are inserted in chronological order as pages in an electronic design notebook". These pages can be marked up and annotated; items of information can be linked to related items on other pages. The result is a personal <u>hyper-document</u> that captures and structures an engineer's knowledge about a project. Selected information can be shared by e-mailing pages to other engineers or to a central project repository, complete with embedded reference pointers and hyper-links. What emerges is an <u>Internet-wide information web</u> that documents and organizes the shared understanding of an entire engineering team [Toye, p. 40, col. 1].

While it is clear that Toye's spatial arrangement of information items on the "NoteMail" page is implemented with a new "Format" data type [e.g., see Toye, p. 40, col. 2, last paragraph], and is therefore different than the prior art Mosaic (APA), Berners-Lee, Raggett I and II combination, the Examiner does not agree with the Patent Owner's sweeping statement that "NoteMail" pages are not even analogous to Web-style hypermedia documents.

However, the Examiner does find the Patent Owner's final argument to be persuasive and dispositive regarding argument III(a):

> 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 11, at paragraphs 36-37].

The Examiner concurs because 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 the data. Therefore, Toye teaches away from the operative coupling between the "executable application" and the "interactive processing" required by the instant '906 patent claims.

Furthermore, Toye teaches that "automatic invoking" of the "appropriate application" is performed by <u>selection</u>, and not by parsing. Toye teaches that notebook data is displayed as a data object or filename <u>that must be selected by a mouse to launch an appropriate application in a separate window</u>" [see Toye page 40, 2nd column, paragraph 2; see also page 36, 2nd column, last paragraph, i.e., " ... ability to construct hyper-documents containing <u>bitmaps</u>, video, and audio"; see also Felten II, at paragraph 47].

Significantly, Toye appears to merely disclose a conventional system for invoking appropriate applications by standard prior art file association techniques, such as invoking the appropriate application based upon the file extension (e.g., when the user clicks and selects a *.doc filename or corresponding file icon and this user action automatically invokes the appropriate word processor). See also Toye: "The functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file" [p. 40, 2nd column, 2nd paragraph].

> *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 response, the Examiner finds the Patent Owner's argument III(b) persuasive for at least the following reasons:

The Examiner notes that the term "dynamic object" is not explicitly used in the Toye disclosure, nor is the term used within the instant '906 claims. It appears the previous Examiner is interpreting the Toye reference to teach the use of an embedded object that is dynamic in the sense that the embedded object may be interactively changed by the user while it is being displayed.

The Examiner concurs and finds dispositive the Patent Owner's argument that the "dynamic objects" taught by Toye are "activated by the user clicking on a static "snap shot" image or icon displayed within a NoteMail page" [see Patent Owner's response, received Oct. 12, 2004, p 22].

The Examiner concurs that the link between the "dynamic object" allegedly taught by Toye and the application to process the "dynamic object" is stored in an external centralized database, and not within the "NoteMail" page itself (as contrasted with the use of the EMBED tag disclosed by Raggett I that provides the link to a rendering application, discussed *supra*; see Raggett I, p. 6).

Accordingly, Toye fails to teach or fairly suggest an ongoing real-time modification or control by a user of a displayed object shown within a browser-controlled window, as performed by an "executable application" that is invoked by parsing an "EMBED" tag to enable "interactive processing" of the type claimed in the '906 patent.

*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.*

In response, the Examiner finds the Patent Owner's argument IV(a) persuasive for at least the following reasons:

"In determining the propriety of the Patent Office case for obviousness in the first instance, it is necessary to ascertain whether or not the reference teachings would appear to be sufficient for one of ordinary skill in the relevant art having the reference before him to make the proposed substitution, combination, or other modification." In re Linter, 458 F.2d 1013, 1016, 173 USPQ 560, 562 (CCPA 1972). The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. In re Mills, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990).

In the rejection set forth on page 6 of the Office Action mailed Oct. 16, 2004, the previous Examiner asserts that the modification of the four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I, and Raggett II, would be motivated based upon "Toye's teaching that its architecture provides openness and flexibility":

> The combination of patentee'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, with distributed hypermedia documents.
>
> 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 (See Toye: p. 40 description of NoteMail, particularly p. 40, col. 2, first paragraph).
>
> It would have been readily apparent to a skilled artisan to modify the method discussed above, combining the teachings of the admitted prior ad in view of Berners-Lee, Raggett I, and Raggett II, by further modifying the combination's static embedded object to be a dynamic embedded object as taught by Toye. **Such a further modification would have been apparent based on Toye's teaching that its architecture provides openness and flexibility** (See Toye: p. 40 col. 2 second complete paragraph).

The support for the "openness and flexibility" motivation relied upon the previous Examiner is taken from the following section of the Toye reference [see p. 40, 2nd column, 2nd and 3rd complete paragraphs]:

> Another 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 onto a notebook page through an embedded "virtual window". When a data object or file is **selected** for inclusion in the

> notebook, the system **will automatically invoke the appropriate application** for displaying that item in the notebook. If the needed application is not locally resident (a likely occurrence in the case of MIME external body references), it will be located and run remotely over the network. Subsequently selecting the displayed data with a mouse will restart the original application, so that the data can be edited or updated without leaving the notebook environment. **The functionality is similar to opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file.** However, applications can now reside anywhere on the Internet.
>
> We are aware of only one other multimedia editor with such an architecture, MediaMosaic [22]. Other engineering notebook projects, by contrast lack this **openness and flexibility**. For example, the Virtual Notebook System [6] can display only static bitmaps; GE's Electronic design Notebook [34], which is built on FrameMaker, can run only those applications whose output formats are compatible with the handful of input formats that FrameMaker accepts.

The Patent Owner argues: "the general and nebulous Toye language regarding 'openness and flexibility' is not related to any possible motivation to combine the references" [see Patent Owner's response received Oct. 12, 2004, p. 23].

In response, the Examiner concurs with the Patent Owner's contention that the "openness and flexibility" motivation applied by the previous Examiner is general and nebulous, for the following reasons:

"Openness and flexibility" is supported by the term "open architecture" in paragraph 2, *supra*, describing a "virtual window" for displaying the output of any application (i.e., suggesting "flexibility") that can display its output through an X-Server. As disclosed by Toye, "any application that displays through an X-server can insert its output (audio, video, or graphics) dynamically onto a notebook page through an embedded 'virtual window' [see Toye, p. 40, 2nd column, paragraph 2]. Toye also teaches a "flexible" system in the sense that if a needed application is not locally resident, it will be "located and run remotely over the network" [see Toye, p. 40, 2nd column, 2nd paragraph].

With respect to the four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II, it is conceded that the section of Toye cited by the previous Examiner would likely provide a motivation to a skilled artisan to modify the four-way combination for the purpose of making it compatible, e.g., with applications that display through an X-Window system, using an X-server.

It is also conceded that, after a user makes a manual selection of a "data object or file," Toye teaches that a local or remote editing application is invoked that can display dynamic objects such as audio and video that may be displayed as an embedded object within a notebook page using the disclosed "virtual window."

However, there is no suggestion to modify the four-way combination to allow a user to interactively process an object embedded within a distributed hypermedia document in accordance with the type of "interactive processing" recited in claims 1 and 6 of the instant '906 patent.

While Toye certainly teaches that the user may select a data object or file and "automatically invoke the appropriate application for displaying that item in the notebook" (as typically performed using file associations in a conventional file manager program) such interactivity (as taught by Toye) <u>can only be initiated by a manual selection performed by the user</u> (i.e., a mouse click or other user selection, as by using a keyboard).

The manual selection step required by Toye defeats the purpose of the use of an EMBED tag that is parsed to invoke an executable application, thus teaching away from the hypothetical four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II.

In contrast, the instant '906 claims require the browser (and not the user) to invoke the "executable application" that in turn executes on the client workstation to enable the claimed "interactive processing."

Accordingly, the Toye reference teaching is insufficient to enable one of ordinary skill in the relevant art having the reference before him to make the proposed substitution, combination, or other modification.

*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.*

"To support the conclusion that the claimed invention is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed invention or the examiner must present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references." Ex parte Clapp, 227 USPQ 972, 973 (Bd. Pat. App. & Inter. 1985).

The Mosaic (APA), Berners-Lee, Raggett I and II combination provides a distributed system where objects may be stored anywhere on the Internet and retrieved using a browser application, e.g., by 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 that uses a single object-oriented database (DIS) to access and store documents.

The Examiner finds the Patent Owner's argument compelling that "the collaborative editing techniques of Toye would be contrary to the publish-and-view philosophy of the Internet." Furthermore, the Examiner concurs that "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" [see Patent Owner's response received Oct. 12, 2004, page 23].

The five-way rejection set forth in the last office action (including the Toye reference) fails to provide a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references.

While Toye does teach dynamic objects (such as audio and video) that may be displayed within the same notebook window using an overlay "virtual window" X-Windows technique, the interactive processing (i.e., editing) taught by Toye <u>can only be invoked manual selection of a data object or file by a user</u> and is therefore not equivalent to the type of interactive processing claimed by the instant '906 patent.

In contrast, the instant '906 claims require the browser (not the user) to invoke the "executable application" that in turn executes on the client workstation to enable the claimed "interactive processing."

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

The "NoteMail" tool combines the functions of an engineering notebook, hypermedia browser and authoring environment, and a file application manager [see Toye, p. 40, col. 1]. The "NoteMail" system is organized in a manner designed to provide maximum benefit to members of a collaborative engineering team.

For example, messages are inserted in chronological order as "NoteMail" pages in an approach that departs from the functionality of prior art web browsers. Prior art web browsers typically organize web page retrieval around stored bookmarks that provide URL links to web pages (and associated objects) that may reside anywhere on the Internet.

The Examiner concurs with the Patent Owner's contention that the "NoteMail" design (i.e., teaching restricted, collaborative access to a centralized database) runs counter to the intended purpose of the Mosaic (APA), Berners-Lee, Raggett I and II hypothetical four-way combination. The intended purpose of the four-way combination is to provide a distributed system that enables universal access to web pages (and associated objects) that may be stored anywhere on the Internet.

In contrast, Toye discloses a system that permits applications to reside anywhere on the Internet, while collaborative, restricted access to the data is only permitted via a centralized database [Toye, p. 40, col. 2, paragraph 2, last line].

Accordingly, the Examiner concurs that the Toye reference teaches away from modifying the Mosaic (APA), Berners-Lee, Raggett I and II hypothetical four-way combination as proposed by the rejection set forth in the last office action.

*PART V. The secondary consideration of commercial success further supports the conclusion of non-obviousness. The attached Declaration of Robert J. Dolan, Dean at the University of Michigan Business School and Gilbert and Ruth Whitaker professor at Michigan Business School ("Dolan") sets forth facts and evidence to legally and factually establish the secondary consideration of commercial success of the invention claimed in claims 1 and 6 of the '906 patent.*

*a. There is a nexus between the claimed invention and the commercial success.*

In response to the Patent Owner's argument V(a), the Examiner has reviewed the supporting "Dolan" Declaration and does not find it persuasive in terms of demonstrating a nexus between the instant claimed '906 invention and commercial success.

The "Dolan" Declaration relies upon the alleged infringement of the '906 patent claims by Microsoft in marketing the Microsoft Internet Explorer browser (IE). In particular, it is alleged that the "IE browser's support for "plug-ins, applets, and Active X functionality incorporates the technology claimed in claims 1 and 6 of the '906 patent" [See "Dolan" Declaration, page 2].

The Patent Owner's argument of commercial success is thus predicated on Microsoft's infringement of the '906 patent as determined by a jury in the trial at the U.S. District Court (Northern District of Illinois, Eastern Division).

However, at the time of this writing, the litigation is still ongoing and is on remand back to the District Court from the CAFC.

While the infringement issue is not being considered on remand from the CAFC, the affirmative defenses of public use and inequitable conduct, if successful, would render the patent invalid and the issue of patent infringement would be moot. Therefore, the PTO does not consider there to be a final judgment on the issue of patent infringement until all appeals have been exhausted and the litigation has concluded. A nexus between the claimed invention and the commercial success of the IE browser cannot be shown (based upon alleged patent infringement) in the absence of a final judgment to establish such infringement.

Accordingly, the Patent Owner has not met the burden of proof required to establish a factual and legally sufficient connection between the evidence of commercial success and the claimed invention such that the evidence is of probative value in the determination of nonobviousness.

*b. The evidence of commercial success is commensurate with the scope of the '906 claims.*

Objective evidence of nonobviousness including commercial success must be commensurate in scope with the claims. In re Tiffin, 448 F.2d 791, 171 USPQ 294 (CCPA 1971). In order to be commensurate in scope with the claims, the commercial success must be due to claimed features, and not due to unclaimed features. Joy Technologies Inc. v. Manbeck, 751 F. Supp.

225, 229, 17 USPQ2d 1257, 1260 (D.D.C. 1990), *aff'd*, 959 F.2d 226, 228, 22 USPQ2d 1153, 1156 (Fed. Cir. 1992).

The Patent Owner relies upon the "Dolan" Declaration (pages 6-9, numbered paragraphs 25-41) to support the contention that the evidence of commercial success is commensurate in scope with claims 1 and 6 of the instant '906 patent.

In response to the Patent Owner's argument V(b), the Examiner need not reach this issue because a nexus between the claimed invention and commercial success has not been established, as discussed in the response to argument V(a), *supra.* A nexus between the claimed invention and the commercial success of the IE browser cannot be shown (based upon alleged patent infringement) in the absence of a final court judgment to establish such infringement (i.e., until all appeals have been exhausted and the litigation has concluded).

The Examiner cannot reasonably address the issue raised by argument V(b) without commenting on the merits of the ongoing litigation. Subject matter concerning patent infringement constitutes a federal question that properly falls within the subject matter jurisdiction of the Federal Court system. Subject matter concerning patent infringement is not considered by the U.S. Patent and Trademark Office.

*c. The commercial success is derived from the invention.*

In response to the Patent Owner's argument V(c), the Examiner does not find the Patent Owner's arguments and the associated "Dolan" Declaration persuasive for the following reasons:

In considering evidence of commercial success, care should be taken to determine that the commercial success alleged is directly derived from the invention claimed, in a marketplace where the consumer is free to choose on the basis of objective principles, and that such success <u>is not the result of heavy promotion or advertising</u>, shift in advertising, consumption by purchasers normally tied to applicant or assignee, or other business events extraneous to the merits of the claimed invention, etc. <u>In re Mageli</u>, 470 F.2d 1380, 176 USPQ 305 (CCPA 1973) (conclusory statements or opinions that increased sales were due to the merits of the invention are entitled to little weight); <u>In re Noznick</u>, 478 F.2d 1260, 178 USPQ 43 (CCPA 1973).

Even assuming, arguendo, that the Patent Owner has demonstrated the required nexus between the instant claimed '906 invention and the commercial success of an allegedly infringing product, the "Dolan" Declaration fails to show that the commercial success of Microsoft's IE browser was not the result of <u>heavy promotion or advertising</u> or other business events extraneous to the merits of the claimed invention.

In particular, Microsoft made the IE browser available to users at little or no cost. Microsoft also bundled the IE browser as an integral component of various Microsoft operating systems (e.g., Windows 95, 98, and Windows

2000). Significantly, the "Dolan" Declaration is silent regarding the issue of free or low cost distribution of the IE browser as a factor in Microsoft's successful capture of market share.

In more traditional business models that involve tangible products, a rational producer will seek to exploit a profit opportunity until the marginal cost of the $n^{th}$ unit produced exceeds the marginal revenue generated from that $n^{th}$ unit. However, when software is distributed over the Internet, the marginal cost of each unit of downloaded software approaches zero as the number of downloads approaches infinity. This is true because the sunk software development costs and the relatively fixed cost of maintaining distribution servers are averaged over a potentially infinite number of downloads.

Obviously, if there exists a quantifiable market demand for a given product, the quantity of units demanded will increase as the cost per unit approaches zero. This was likely true in the case of the Microsoft IE browser because it was offered to the public as a free download (or merely for the cost of the CD media plus postage and handling).

Microsoft clearly offered the IE browser to the public at little or no cost in an effort to gain market share over the competing Netscape browser, even though it may also be true that Microsoft viewed the functionality of Active X (allegedly infringing upon the '906 patent functionality) as giving IE an advantage over Netscape [e.g., see "Dolan" Declaration, page 8, paragraph 36]. In addition, such free distribution of the IE browser clearly promoted and helped to advertise Microsoft's main operating system and application software products.

Because Microsoft made the IE browser available to the public at little or no cost, the past distribution of IE has at least the appearance of "heavy promotion or advertising." While the alleged infringement of '906 functionality may indeed have been a factor in the market success of the IE browser, patent infringement has not been shown by a final court judgment. Significantly, the Patent Owner has failed to address the Microsoft marketing strategy of distributing the IE browser to the public at little or no cost.

Because Microsoft was already an established market leader with respect to desktop operating systems and applications, the success of the IE browser could also be reasonably attributed to Microsoft's extensive advertising and position as a market leader before the introduction of the allegedly infringing product (i.e., the IE browser). See Pentec, Inc. v. Graphic Controls Corp., 776 F.2d 309, 227 USPQ 766 (Fed. Cir. 1985) (commercial success may have been attributable to extensive advertising and position as a market leader before the introduction of the patented product).

Accordingly, even when the facts are viewed in a light most favorable to the Patent Owner, the Patent Owner has failed to demonstrate by a preponderance of the evidence that the commercial success of Microsoft's IE browser was derived from the instant '906 invention.

## The Viola Code

The CAFC opinion (Docket No. 04-1234, March 2, 2005) states on page 11, 2nd paragraph:

> In contrast, the record indicates Wei not only demonstrated DX34 to two Sun Microsystems engineers without a confidentiality agreement (on May 7, 1993), but only twenty-four days later (on May 31, 1993) posted DX37 on a publicly-accessible Internet site and notified a Sun Microsystems engineer that DX37 was available for downloading.

The '906 invention was reduced to practice no later than January, 27, 1994 when it was presented on that date to a conference "Medicine Meets Virtual Reality II." [2] From the court record, it is clear that the date of publication on the Internet of the DX37 code (May 31, 1993) antedates the date of reduction to practice (Jan. 27, 1994) of the '906 invention. Accordingly, the DX37 code submitted by the Patent Owner on Dec. 30, 2003 (received by the PTO on Jan 5, 2004) has been considered by the Patent and Trademark Office as a publication that constitutes prior art for purposes of this reexamination proceeding.

The "Viola Code" is stored as an artifact (i.e., a CD disk) associated with the instant Image File Wrapper (IFW) reexamination file. The contents of artifacts are not stored as images on the PTO IFW system. The Viola code CD contains two compressed zip files representing "Viola Source code" ("DX34" and "DX37"):

[2] See "Ruling on the Defense of Inequitable Conduct", No. 99 C 626, U.S. District Court Northern District of Illinois, Eastern Division, page 9.

1) viola930512.tar.gz.zip - this compressed file represents the earlier Viola source code, also referred to as "DX34" in the CAFC opinion (Docket no. 04-1234, March 2, 2005, see also IFW "Reexam Notice of Court Action" dated April 11, 2005; see especially page 11 as numbered in the printout (corresponding to IFW page 16 of 32). The viola930512.tar.gz.zip (i.e., "DX34") file, when unzipped, contains 1,027 files in 35 folders consisting of 8 total megabytes in size.

2) violaTOGO.tar.Z.zip - this compressed file represents the later Viola source code, also referred to as "DX37" in the CAFC opinion. The violaTOGO.tar.Z.zip (i.e., "DX37") file, when unzipped, contains 1,030 files in 34 folders consisting of 7.7 total megabytes in size.

To conduct a thorough and comprehensive review of the DX37 code (1,030 files), the Examiner successfully unzipped the provided "violaTOGO.tar.Z.zip" compressed file and indexed all DX37 files using a commercially available text searching program designed for such purpose. [3]

In this manner, every DX37 file containing textual content (including code) was fully and comprehensively text searched with the resulting "hits" being highlighted in the full-text context of each document. Several representative Viola files are reproduced *infra* to clarify the scope of the Viola DX37 prior art publication.

[3] The Examiner used the "dtSearch" program to index and text search all DX7 files that contained textual content. See **http://www.dtsearch.com/**

**How Viola embeds Viola scripts in a hypermedia document**

In particular, the file "violaApps.hmml" (contained in the "docs" directory) illustrates how interactive applications (i.e., actually Viola scripts) are embedded in a Viola hypermedia document as designated by a matched pair of <VOBJF> and </VOBJF> tags that specify a Viola script that is used to generate the embedded object, as shown below:

```
<VOBJF> ../apps/clock.v </VOBJF>
```

When the Viola hypermedia browser parses the hypermedia document (e.g., "violaApps.hmml", denoting a hypermedia document written in Hyper Media Markup Language) and encounters the matched pair <VOBJF> and </VOBJF> tags, the browser then retrieves the Viola script "clock.v" from the directory location specified by the directory path (i.e., ../apps/ ).

Significantly, the Viola script "clock.v" is INTERPRETED to embed an interactive application object within the same window of the Viola browser. Each Viola script line is interpreted by translating the Viola script code (or corresponding byte code) to native binary machine code instructions that are executed in a sequential fashion.

The Viola documentation states: "The extension language is C-like in syntax and is processed into byte-code for efficient interpretation" [see "violaCh1.hmml" in the "docs" directory]. Although the aforementioned "clock.v" example is clearly a Viola script, it appears that an intermediate byte-code representation may be interpreted at runtime. In such case, the Viola script must be compiled in advance to intermediate byte-code form.

The "violaApps.hmml" hypermedia document file (as parsed by the Viola browser) and the corresponding "clock.v" script file are shown below:

"violaApps.hmml" illustrating the use of the Viola <VOBJF> object tags (located within the "docs" directory)

```
<!DOCTYPE hmml SYSTEM>
<TITLE>Test</TITLE>
<H1>List No. 5</H1>
<P>
The <CMD>&lt;VOBJF&gt;</CMD> tag can be used to insert viola
applications.
Using this capability allows you embed in your document what you
can access or build using viola's programming, and GUIs. Of
course too much violaism reduces the portability of your document
on the World Wide Web,but anyway...
<P>
Here are some examples.
<H2>Clock</H2>
<VOBJF>../apps/clock.v</VOBJF>
<H2>Vicon</H2>
<VOBJF>../apps/vicon.v</VOBJF>
<P>
This can be a handy menu to tuck away at a corner of the screen.
<H2>Query</H2>
<VOBJF>../apps/vwq.v</VOBJF>
<P>
This application is intended to gather user information.
<H2>Wave fun</H2>
<VOBJF>../apps/wave.v</VOBJF>
<H2>Noodle Doodles</H2>
<VOBJF>../apps/doodle.v</VOBJF>
<P>
So I was bored...
<P>
The end.
```

The first portion of the corresponding "clock.v" Viola script (located in the "apps" directory)

```
\name {clock}
\class {vpane}
\parent {}
\width {200}
\height {210}
\children {clock.dial clock.mesg}
```

```
\
\name {clock.dial}
\class {XPMBG}
\parent {clock}
\script     {
      print("@@@@@@ clock: ");
      for (i =0; i < arg[]; i++) print(arg[i], ", ");
      print("\n");

      switch (arg[0]) {
      case "tick":

            date = date();
            clock.mesg("update");
            second = int(nthWord(date, 6));
            minute = int(nthWord(date, 5));
            hour = int(nthWord(date, 4));
            if (hour >= 12) hour = hour - 12;

            secondD = (second / 60.0 * 360.0) - 90.0;
            minuteD = (minute / 60.0 * 360.0) - 90.0;
            hourD = (hour / 12.0 * 360.0) - 90.0 + (minute / 60.0 * 30.0);

            secondX = secondR * cos(secondD) + centerX;
            secondY = secondR * sin(secondD) + centerY;
            minuteX = minuteR * cos(minuteD) + centerX;
            minuteY = minuteR * sin(minuteD) + centerY;
            hourX = hourR * cos(hourD) + centerX;
            hourY = hourR * sin(hourD) + centerY;

            if (lminuteX != minuteX) {
                  clearWindow();
                  clock.dial("render"); /* brutally redraw */
                  drawLine(centerX, centerY, minuteX, minuteY);
                  drawLine(centerX, centerY, hourX, hourY);
                  invertLine(centerX, centerY, lsecondX, lsecondY);
            }
            invertLine(centerX, centerY, lsecondX, lsecondY);
            invertLine(centerX, centerY, secondX, secondY);

            lsecondX = secondX;
            lsecondY = secondY;
            lminuteX = minuteX;
            lminuteY = minuteY;
            lhourX = hourX;
            lhourY = hourY;
            if (view) after(1000, "clock.dial", "tick");
            return;
      break;
      case "render":
            usual();
            for (i = 1; i <= 12; i = i + 1) {
```

```
                        x = letterR * cos((i / 12.0 * 360) - 90) +
                              centerX - 10;
                        y = letterR * sin((i / 12.0 * 360) - 90) +
                              centerY - 5;
                        drawText(x, y, i, str(i));
                  }
                  return;
            break;
            case "VIEW_ON":
                  view = 1;
                  return;
            break;
            case "VIEW_OFF":
                  view = 0;
                  return;
            break;
            case "expose":
                  clearWindow();
                  lminuteX = 0; /* forces redrawing */
                  lhourX = 0; /* forces redrawing */
            break;
            case "config":
                  usual();
                  send(self(), "resize", arg[3], arg[4]);
                  return;
            break;
            case "resize":
                  if (arg[1] < arg[2])
                        radius = arg[1] / 2.0;
                  else
                        radius = arg[2] / 2.0;

                  centerX = arg[1] / 2.0;
                  centerY = arg[2] / 2.0;
                  secondR = radius * 0.95;
                  minuteR = radius * 0.9;
                  hourR       = radius * 0.6;
                  letterR = (radius - 9) * 0.94;

                  after(2000, "clock.dial", "tick");
                  lminuteX = 0;

/*                system(concat(environVar("VIOLA"), "/play ",
                              environVar("VIOLA_DOCS"), "/cuckoo.au"));
*/
            break;
            }
            usual();
}
```

The Viola DX37 approach to embedding interactive objects using interpreted Viola scripts (or corresponding byte-code forms) does not anticipate nor fairly suggest the '906 invention as claimed for at least the following reasons:

While Viola DX37 supports hypermedia and a type of interpreted script-based interactive processing, the Examiner can find no indication from a comprehensive text search of the Viola DX37 files that such interactivity results from the use of a parsed **embed text format** that specifies the location of an **object** external to the hypermedia document, where the browser application **uses type information associated with the object to identify and locate an external executable application**, and where the parsing step results in the browser automatically invoking the **executable application** to display the **object** and enable interactive processing of the **object** within the same browser-controlled window, when the instant '906 patent claims 1 and 6 are properly accorded the broadest reasonable interpretation consistent with the specification.

## I. VIOLA <VOBJF> TAGS DO NOT ANTICIPATE NOR FAIRLY SUGGEST THE "EMBED TEXT FORMAT" AS CLAIMED IN THE '906 PATENT.

Unlike the instant '906 claimed "embed text format," the Viola <VOBJF> tags use no arguments or additional elements beyond a directory path and filename. The Viola <VOBJF> tag simply loads the Viola script using the path and filename specified between the <VOBJF> and </VOBJF> tags, as shown:

```
<VOBJF> ../apps/clock.v </VOBJF>
```

In contrast, the browser application of the instant '906 patent uses a type element associated with the external object (i.e., "type information" as claimed) to identify and locate an executable application external to the distributed hypermedia document [see '906 patent, TABLE II and associated discussion col. 13].

Significantly, the Viola browser application <u>does not fairly teach nor suggest where the browser application uses type information associated with the external object to identify and locate an external executable application</u>.

## II. VIOLA SCRIPTS (OR CORRESPONDING BYTE-CODE FORMS) DO NOT ANTICIPATE NOR FAIRLY SUGGEST THE EXTERNAL "OBJECT" AS CLAIMED IN THE '906 PATENT.

If the Viola <VOBJF> tags are considered as arguably corresponding to the instant claimed '906 "embed text format" (in the sense that the Viola <VOBJF> tags specify "the location of at least a portion of an object external to the first distributed hypermedia document" as claimed in '906 claims 1 and 6), then the Viola script program specified between the <VOBJF> tags is not equivalent to the instant '906 claimed external "object" when the claimed '906 external "object" is interpreted in a manner <u>consistent with the specification of the '906 patent</u>.

The Viola, "clock.v" script is a high-level source code PROGRAM. In contrast, the scope of the claimed '906 external "object" broadly encompasses myriad types of <u>data objects</u>, including <u>self-extracting data objects</u> [see '906 patent, col. 3, lines 33-51].

The scope of the claimed '906 external "object" is broad when construed in a manner consistent with the specification (i.e., see '906 patent, col. 3, lines 36-39: "a data object is information capable of being retrieved and presented to a user of a computer system."). However, the scope of the claimed '906 external "object" clearly does not read upon a high-level source code PROGRAM, such as a Viola script, nor does it read upon an object in byte-code form.

When the scope of the claimed '906 external "object" is construed in a manner consistent with the specification, it is clear that any executable component of the claimed '906 external data "object" is limited to performing self-extraction of the compressed data object:

See '906 patent, col. 3, lines 43-51:

> When a browser retrieves an object such as a self-extracting **data object** the browser may allow the user to "launch" the self-extracting **data object** to automatically execute the unpacking instructions to expand the data object to its original size. Such a combination of executable code and data <u>is limited in that the user can do no more than invoke the code to perform a singular</u>

<u>function such as performing the self-extraction after which time the object is a standard data object</u>.

Although a self-extracting data object typically includes executable code to expand the compressed data object to its original size, this type of self-extraction extracts DATA that has no relationship to a high-level source code PROGRAM in the form of a Viola script, or a byte-code file, or the like.

## III. VIOLA SCRIPTS (OR CORRESPONDING BYTE-CODE FORMS) DO NOT ANTICIPATE NOR FAIRLY SUGGEST THE EXTERNAL "EXECUTABLE APPLICATION" AS CLAIMED IN THE '906 PATENT.

The Examiner finds that the Viola code publication does not fairly teach nor suggest that the browser automatically invokes an **executable application**, external to the hypermedia document, to display the object and enable interactive processing of the object, when the instant '906 patent claims 1 and 6 are properly accorded the broadest reasonable interpretation <u>consistent with the specification</u>, where such interpretation is also <u>consistent with the interpretation that those skilled in the art would reach</u>. <u>In re Hyatt</u>, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000); <u>In re Cortright</u>, 165 F.3d 1353, 1359, 49 USPQ2d 1464, 1468 (Fed. Cir. 1999).

While expert witnesses and dictionaries (considered as extrinsic evidence) may differ regarding the proper construction of the instant claimed "executable application", the Central Processing Unit (i.e., CPU or microprocessor) found in every computer system has only a single, precisely defined interpretation as to what constitutes an "executable application." When the CPU initiates a "fetch and execute" cycle, the program counter is loaded with the address of the next executable instruction. To be "executable" the contents of the memory location pointed to by the program counter must contain an instruction in binary form that is a member of the native instruction set of the microprocessor (i.e., a binary machine language instruction). The binary representation of the precise portion of the machine language instruction that determines what kind of action the computer should take (e.g., add, jump, load, store) is referred to as an operation code (i.e., OP code). From the perspective of the CPU, if a recognizable machine language instruction (i.e., a native CPU instruction) is not found within the memory location pointed to by the program counter, the computer will crash.

The Viola system uses "C-like" Viola scripts that must be INTERPRETED by the browser and then TRANSLATED or CONVERTED into binary native executable machine code that can be understood by the CPU. Alternately, the Viola script is precompiled to intermediate byte-code form and the byte-code is interpreted (i.e., translated) into binary native executable machine code at runtime. This extra step of translation results in an unavoidable performance penalty, as interpreted applications run much slower than compiled native binary executable applications.

Accordingly, the "C-like" Viola scripts (or corresponding byte-code representations) are not "executable applications" from the perspective of the CPU, which is the only perspective that really matters at runtime. A conventional CPU is only capable of processing binary machine language instructions from its own native instruction set.

Without an intermediate translation step performed by an interpreter component of the Viola browser, a Viola script (or corresponding byte-code representation) cannot be processed as an executable application by the CPU.

Significantly, the instant '906 specification is silent regarding the use of applications that rely upon scripts that must be interpreted before they can be executed. The instant '906 specification is silent with respect to interpreting code prior to execution. The instant '906 specification is silent with respect to the use of byte-code intermediate forms.

## IV. THE INTENDED USE OF THE VIOLA RAPID PROTOTYPING INTERPRETED SCRIPTING SYSTEM TEACHES AWAY FROM THE INTENDED USE OF THE '906 PATENT.

The Viola scripting system teaches away from the primary intended use of the '906 invention. The main object of the Viola scripting system was to provide an <u>interpreted</u> operating environment <u>primarily designed for rapid prototyping</u>.

In contrast, the main object of the '906 invention is to provide a system "that allows the accessing, display and manipulation of large amounts of data, especially image data, over the Internet to a small, and relatively cheap, client computer ['906 patent, col. 6, lines 21-25].

The use of an interpreted script application (or corresponding intermediate byte-code representation) in the '906 patent context would be unacceptably slow in processing large amounts of data, especially the kind of complex three-dimensional image data used in one embodiment of the '906 patent. One must reflect on the fact that the personal computers used in 1994 were significantly slower than the high speed computers widely used today (2005).

Overcoming the existing bandwidth and processing speed constraints associated with the prior art are central objects of the '906 invention [see '906 patent, col. 5, lines 39-56]:

> The open distributed hypermedia system provided by the Internet allows users to easily access and retrieve different data objects located in remote geographic locations on the Internet. However, this open distributed hypermedia system as it currently exists **has shortcomings** in that today's large data objects are limited largely by **bandwidth constraints** in the various communication links in the Internet and localized networks, and by the limited processing power, or computing constraints, of small computer systems normally provided to most users. **Large data objects are difficult to update at frame rates fast enough (e.g., 30 frames per second) to achieve smooth animation.** Moreover, **the processing power needed to perform the calculations to animate such images in real time does not exist on most workstations, not to mention personal computers.** Today's browsers and viewers **are not capable of performing the computation necessary to generate and render new views of these large data objects in real time.**

Also see '906 patent, col. 6, lines 21-31:

> On the other hand, small client computers in the form of personal computers or workstations such as client computer 108 of FIG. 2 are generally available to a much larger number of researchers. Further, it is common for these smaller computers to be connected to the Internet. **Thus, it is desirable to have a system that allows the accessing, display and manipulation of large amounts of data, especially**

> **image data, over the Internet to a small, and relatively cheap, client computer.**
>
> Due to the relatively **low bandwidth of the Internet** (as compared to today's large data objects) and the **relatively small amount of processing power available at client computers**, many valuable tasks performed by computers cannot be performed by users at client computers on the Internet.

The importance of "speed of access" to application client 210 (corresponding to the instant claimed "executable application") is further demonstrated by the use of "Terminate and Stay Resident" (TSR) programs to provide faster access [See '906 patent, col. 8, lines 66, 67, cont'd, col. 9, lines 1-14]:

> Client computer 200 includes processes, such as browser client 208 and **application client 210**. In a preferred embodiment, **application client 210** is resident within client computer 200 prior to browser client 208's parsing of a hypermedia document as discussed below. In a preferred embodiment application client 210 resides on the hard disk or RAM of client computer 200 and is loaded (if necessary) and executed when browser client 208 detects a link to **application client 210**. The preferred embodiment uses the XEvent interprocess communication protocol to exchange information between browser client 208 and **application client 210** as described in more detail, below. Another possibility is to install **application client 210** as a **"terminate and stay resident" (TSR) program** in an operating system environment, such as X-Window. Thereby making access to **application client 210** much faster.

The Examiner submits that "Terminate and Stay Resident" (TSR) programs were notoriously understood to be native binary executable code by those of ordinary skill in the art at the time of the '906 invention. [4]

For example, in the legacy Microsoft MS-DOS environment, TSR programs were native binary executables designated as COM or EXE programs that were preloaded in memory for fast execution. TSR programs were typically used to allow utilities, drivers, or interrupt handlers to be preloaded in memory for quick access. [5] The purpose of memory preloading for quick access would not be well served if a TSR program in the form of a script had to be interpreted (i.e., translated) to binary native code before it could be executed.

In addition, the '906 patent teaches the use of applications such as "spreadsheet programs, database programs, and word processor programs" [see col. 13, line 14]. The Examiner submits that at the time of the invention most commercial spreadsheet programs, database programs, and word processor applications were usually sold as native binary executable applications. The Examiner does concede that applications of the aforementioned types were available in interpreted languages at the time of the invention (e.g., a database program written in the BASIC language). However, an interpreted application in source code form cannot be executed directly by the CPU without first being translated to native binary executable machine code form, as discussed *supra*.

[4] See e.g., U.S. Patent 5,056,057 to Johnson et al., "Keyboard interface for use in computers incorporating **terminate-and-stay-resident** programs", issued Oct. 8, 1991.
[5] Duncan, Ray, "Advanced MSDOS Programming", Microsoft Press, 1986, page 391.

## V. EVEN ASSUMING, ARGUENDO, THAT "INTERPRETING A SCRIPT" (OR CORRESPONDING BYTE-CODE REPRESENTATION) MAY BE BROADLY CONSIDERED AS EQUIVALENT TO "EXECUTING AN APPLICATION", SUCH INTEPRETATION MERGES THE BROWSER AND THE "EXECUTABLE APPLICATION" INTO ONE PROGRAM THAT FAILS TO TEACH EVERY ELEMENT OF THE '906 PATENT CLAIMS.

Assuming *arguendo* that one adopts the alternate broader modern construction where "interpreting a script" (or interpreted the corresponding byte-code representation) may be considered as equivalent to "executing an application," then the Viola script arguably becomes an integral component of the Viola browser that parses, interprets (i.e. translates), and executes each line of the script (or corresponding byte-code). In such case, the browser and the "executable application" merge into one program, and therefore cannot meet the requirement for a discrete "browser application" and a discrete "executable application" as claimed by the instant '906 patent [see claims 1 and 6].

Lastly, The Examiner takes particular note of the fourth line of the "violaBrief.hmml" file ("Technical Overview of Viola," see the "docs" directory) that leads one to conclude that the Viola DX37 invention may not have been fully enabled at the time of publication:

```
          <TITLE>Viola, A Technical Summary</TITLE>
     <CAUTION>THIS DOCUMENT IS IN DRAFT STATUS</CAUTION>
```

For at least the aforementioned reasons, the DX37 Viola files, when considered as a prior art publication for purposes of reexamination, do not teach nor fairly suggest the instant '906 invention, as claimed.

An appendix is attached that presents some of the more relevant Viola documentation files. The files were created for display by a Viola browser and are presented with the included hypermedia tags as found on the CD artifact disk.

## <u>Conclusion</u>

In summary, the Examiner concurs with the Patent Owner with respect to arguments I-IV for the reasons discussed *supra*.

Although the Examiner does not concur with the Patent Owner with respect to argument V, the issue of establishing a nexus between the claimed invention and commercial success is not dispositive.

The Patent Owner's arguments traversing the rejection need only prevail by the "preponderance of the evidence" standard to succeed in having the rejections set forth in the last office action withdrawn. The ultimate determination of patentability must be based on consideration of the entire record, by a preponderance of evidence, with due consideration to the persuasiveness of any arguments and any secondary evidence. <u>In re Oetiker</u>, 977 F.2d 1443, 24 USPQ2d 1443 (Fed. Cir. 1992).

Accordingly, for at least the aforementioned reasons set forth with respect to arguments I-IV, the Examiner has reconsidered and withdrawn the rejections set forth in the last office action (mailed Aug. 16, 2004).

In addition, the DX37 Viola files have been considered as a prior art publication. For the reasons discussed *supra*, the DX37 Viola files included on the CD artifact do not teach nor fairly suggest the instant '906 invention, as claimed.

Instant U.S. Patent 5,838,906 claims 1-10 are hereby confirmed.

Any comments considered necessary by PATENT OWNER regarding the above statement must be submitted promptly to avoid processing delays. Such submission by the patent owner should be labeled: "Comments on Statement of Reasons for Patentability and/or Confirmation" and will be placed in the reexamination file.

St. John Courtenay III
Primary Examiner
Central Reexamination Art Unit 3992

Mark Reinhart, First Conferee
Special Programs Examiner (SPRE)
Central Reexamination Art Unit 3992

Second Conferee

## VIOLA APPENDIX

The contents of file "viola.desc" (contained in the "docs" directory):

```
language:       viola (Visual Interactive O Language and Application)
package:        viola
version:        2.0 beta
parts:          interpreter, applications, documentation,
how to get:     ftp xcf.berkeley.edu src/local/viola/*
description:    A language/toolkit for hypermedia applications. Very loosely
                modeled after Hypercard. Intended as a tool for building
                and running hypermedia applications that are composed of
                collection of classed objects interacting with the user
                and passing messages among each other. Has simple GUI
                specification language. Is event driven (X-Window, timer,
                I/O). Notion of "objects" for modularization and scalability.
                Syntax is C like. Bytecode compilation is done incremental
                (object by object). Is single class inheritanced.
                + has objects
                + dynamic array
                + message passing
                + bytecode compiler, interpreter
                + graphical interface toolkit
                + pseudo-terminal I/O interface
                + socket I/O interface
                + world wide web interface
                - non dynamic class definition (C level definition)
                - non dynamic data types: string, char, int, float, array
                - little interactive authoring tools for naive users
                - development on language is slow, but application driven
ports:          Unix/X
author:         Pei Y. Wei <wei@xcf.berkeley.edu>
status:         actively developed, application driven
discussion:     viola@xcf.berkeley.edu
updated:

-----------
reference:      The X Resources, O'Reilly & Associates

    europe:     ftp info.cern.ch pub/www/src/viola*
    japan:      ftp srawgw.sra.co.jp pub/x11/viola*
```

The file "violaBrief.hmml" (contained in the "docs" directory) provides a technical overview of Viola, and is reproduced for the record in its entirety below (including the "hmml tags" associated with the browser hypermedia markup language):

"violaBrief.hmml" Technical Overview of Viola (see the "docs" directory)

```
<!DOCTYPE hmml SYSTEM>
<HMML>
<TITLE>Viola, A Technical Summary</TITLE>
<CAUTION>THIS DOCUMENT IS IN DRAFT STATUS</CAUTION>
<HSEP>gold</HSEP>
<H1>VIOLA</H1>
<H3>Visual Interactive Object-oriented Langage and Applications</H3>
<P>
This paper presents a technical overview of viola.

<HSEP>gold</HSEP>
<H1>Overview</H1>
<P>
Viola is a tool for the development and support of interactive media
applications. Its basic functionality is not unlike that of
HyperCard and Tcl/Tk. Viola uses an object oriented model for
encapsulating data into ``object'' units, and to enforce a
classing and inheritance system. The extension language is
C-like in syntax, and is compiled into byte-code for
efficient interpretation. The graphical elements (widgets)
exist as classes in the Viola class hierarchy. The set of
widgets implemented in Viola are similar to those found in
graphical user interface toolkits like Xt, plus more
unusual widgets such as HyperCard-like cards and invisible
celopane buttons, and hypertext textfield.

<H2>Classes and Objects</H2>
<P>
The single inheritance classing system defines the basic types of
object instances. Many of these predefined class types happen to be
GUI oriented, because of the current application emphasis on hypermedia,
but many are non-visual and have nothing to do with GUIs. A modular object
model is enforced to control complexity: to provide a relatively simple
way of data encapsualization; for improving the size scalability of viola
applications; and for possibly helping network distribution of objects.
A scripting language exists for application writers to program
modifications to default object behaviors, and for application programming.
<P>
This is the Viola class hierarchy as of this writing.
It is rapidly evolving:
<VOBJF>../apps/chier.v</VOBJF>
```

```
<P>
This class hierarchy seems deficient (at this point and from
some point of view, it's probably true) compared to the GUIs
provided by toolkits like Motif. But, it's actually not as
deficient as it seems. For the same reason as Tk, Viola does
not require hard coding of, for example, dialog boxes to
achieve the same functionality.
<P>
Because of the interpretive nature of the system, complex
GUIs can be composed out of primitive elements, dynamically.
To build a dialog box, a script could be written to create
and necessary objects, and somehow combine them together to
constitute a dialog box.
<P>
Making a dialog box can be made easy by calling a pre written procedure.
The current way to do this in Viola is to build a ``dialog box maker
object'', and to send to it a message:
``Please make me a dialog box, with the following specifications''.

<H3>Hello, World!</H3>
<P>
Here's the proverbial <ITALIC>Hello World</ITALIC> program.
Go ahead, click on it.
<VOBJF>../apps/violaBriefExample_hello.v</VOBJF>
<P>And its file level representation:
<EXAMPLE>
\class {txtButton}
\name {violaBriefExample_hello}
\label {Hello, world!}
\script {
        switch (arg[0]) {
        case "buttonRelease":
                bell();
        break;
        }
        usual();
}
\width {100}
\height {30}
\BGColor {grey45}
\BDColor {white}
\FGColor {white}
\
</EXAMPLE>
<P>
In reality the <CMD>switch()</CMD> would be busier than just handling
one message. But it could just as easily be written thusly (and with
non-essential color information left out):
<EXAMPLE>
\class {txtButton}
\name {hello}
\label {Hello, world!}
```

```
\script {
	if (arg[0] == "buttonRelease") bell();
	usual();
}
\width {100}
\height {30}
</EXAMPLE>
```

<P>
Although it may seem that some simple binding mechanism would be less verbose, this free form allows one to easily compose the message handler in any order -- doing the default action first, then do the special thing, or any which way.

<H2>Messaging system</H2>
<P>
Viola is message driven, and messages may be generated by a number of sources. A message is typically caused by the user interacting with a graphical user interface object, but it could also be generated by other objects, or by a timer facility. Through a communication facility such as the socket, a message may also be generated from another process on the network.
<P>
In the above ``Hello, world!'' example, when the button is clicked on, that button object "hello" will eventually receive a "buttonRelease" message, which according to the script will execute the <CMD>bell()</CMD> command. If the object does not have any message handlers, the message will ``fall thru'' the object, and (by way of <CMD>usual()</CMD>) the class default action will occur.
<P>
A typical viola application consists of a collection of objects interacting -- generating, receiving, and delegating messages -- with each other, and with the user.

<H2>The Extension Language</H2>
<P>
As seen in the example above, viola scripts are C-like in syntax. The language supports way few constructs: <CMD>if, while, for, switch</CMD>. The commands like <CMD>print(), exit(), create()</CMD>, etc are all implemented as <ITALIC>methods</ITALIC>. Instead of building the commonly used commands into the language grammar, they actually are just defined early enough in the class hierarchy as to be accessible by all subclasses that may need them.
<P>
All objects can be individually programmed using the scripting language. Each object is essentially its own interpretive environment, and each object is its own variable scope.
<P>
For optimization, object scripts are compiled into <ITALIC>byte codes</ITALIC> before applying the byte code interpreter on them. Because an object's script is basically a message event handler that is likely to receives many messages in its instance life time, the one time

```
cost of parsing and simple transformation into byte codes is
very worthwhile. The gain in execution speed is especially
apparent when the objects deal with time critical "mouseMove"
messages, or if there are tight looping operations.

<HSEP>gold</HSEP>
<H1>Applications</H1>
<P>
Along side the development of the Viola language/toolkit
<ITALIC>engine</ITALIC> itself, there is also the development of real
working applications using the engine. The two processes provide reality
checks for each other.
<P>
Here we show screendumps of two developing viola applications.

<H2>World Wide Web Browser</H2>
<FIGURE TYPE="image/gif" SRC="../docs/violaWWW.gif">
<P>
This ``ViolaWWW'' application is currently among the most actively developed
viola application. The initial viola-WWW effort was made in order to
provide to viola a clean network transport mechanism. But the ViolaWWW
browser application itself turned out to be useful enough, that it is being
actively developed, with emphasis on support for online publishing.
<P>
An early version of this browser has been in use in the WWW community
since mid 92, it being the first publically available World Wide Web
browser for X-Windows.

<H2>The Whole Internet Resource Catalog</H2>
<FIGURE TYPE="image/gif" SRC="twi.gif">
<P>
An electronic version of the resource catalog portion of the book
<ITALIC>The Whole Internet</ITALIC>. This application uses HyperCard
style card-flipping technique to flip among four basic GUI sets
(the cover frame is shown here; others frames contain documents and
controlling GUI elements).

<HSEP>gold</HSEP>
<H1>Summary</H1>
<P>
In sum, the Viola language/toolkit system provides an environment
where applications are composed of groups of objects, where objects
interact, by message passing, with the user and with each other.
<P>
As more applications are developed, more reusable objects will be created.
And development of successive applications will become easier and easier.
One of the goals of the Viola project is to accumulate a collection of
objects useful for constructing hypermedia applications.
<P>
The immediate future direction of Viola development will continue to aim
towards the path of hypermedia applications, with the World Wide Web
as the document/object network transport infrastructure.
```

```
<P>
If you're interested in contributing to the development effort,
please contact me.
<HSEP>gold</HSEP>
<ADDRESS>
<P>Pei Y. Wei
<P>Developer, O'Reilly & Associates, Digital Media Group
<P><CMD>wei@ora.com</CMD>
</ADDRESS>
</HMML>
```

The contents of file "violaCh1.hmml" (contained in the "docs" directory):

```
<!DOCTYPE hmml SYSTEM
[
]>
<HMML>
<SECTION NAME="chapter1">
<H1>Introduction to Viola</H1>
<FIGURE TYPE="image/xbm" SRC="viola.xbm">

<SECTION NAME="whatIsViola">
<H2>What is Viola?</H2>
<P>
Viola is a hypermedia application authoring and supporting system.
It contains a graphical user interface set, an ``object oriented''
data organization and storage model, and a built-in extension
language. Perhaps the most important contribution of Viola is its
potential in bringing HyperCard-like capability to a very wide
range of platforms.
<p>
Viola can be used for the development and support of
interactive media applications. It provides an object data
organization model, an interpreted extension language,
graphical elements for user interface. The Viola operating
environment is interpretive, designed for rapid prototyping.
<P>
Viola is desigend to aid the development and support of
interactive/hyper media applications for the Unix/X platform.
Its functionality is similar to HyperCard and Tcl/Tk.
Viola uses an object oriented model to facilitate data
encapsulation into ``object'' units, and to enforce a
 classing and inheritance system. The extension language is
C-like in syntax and is processed into byte-code for
efficient interpretation. The graphical elements (widgets)
exist as classes in the Viola class hierarchy. The set of
widgets implemented in Viola are similar to those found in
user interface toolkits like Xt, plus more unusual widgets
such as HyperCard-like cards and invisible celopane buttons,
and hypertext textfield.
```

```
<P>
In sum, Viola provides an environment in which applications
are composed of groups of objects where each object interacts,
by message passing, with the user and with each other.
<P>
Because most aspects of an object is accessible and controllable
through the interpreted extension language, building an
application in Viola can be done dynamically, without the
edit/compile cycle. As with other systems with built-in
extension language (Emacs/ELisp, Tk/Tcl, HyperCard/HyperTalk),
Viola derives much of its versitility from its extension
language.
<P>
The rest of this paper gives a brief overview of the Viola
basics: the object model, language, and GUI elements.
It also describes some applications(?).
</SECTION>

<SECTION NAME="objectSystem">
<H2>The Object System</H2>
<P>
This section briefly describes Viola's notion of object
orientation.
<P>
Each Viola object consists of an array of ``slot'' values.
These values are information pertaining specifically to the
object: its class, name, script, color, and so on. The number
and type of each slot in an object are determined by the class
of the object.
<P>
Each class inherits slot definitions from its superclasses,
and has the option to set new values for the inherited slots.
In addition to those inherited slots, it may define two types
of new slots: private and common.
<P>
Common slots define slots that are shared by all object
instances of the same class. Private slots define slots that
make up each object instance. The separation of common and
private slots reduces redundancy of information carried by
each object.
<P>
As with slots, class methods are also inherited. The idea,
again, is to provide a mechanism for sharing as much code
as possible. It also makes the task of subclasing relatively
easy and systematic. It should be noted that modification of
the object system (to subclass, adding slots and methods)
must, at this point, be done in C.
<P>
This is the Viola class hierarchy as of this writing. It is
evolving rapidly.
</SECTION>
```

```
<SECTION NAME="violaClassHierarchy">
<H2>The Viola Class Hierarchy</H2>
<EXAMPLE>
        cosmic
                generic
                        field
                                BCard
                                FCard
                                XBM
                                        XBMButton
                                        toggle
                                XPM
                                        XPMButton
                                GIF
                                dial
                                client
                                        TTY
                                        socket
                                menu
                                pane
                                        hpane
                                                txt
                                                txtLabel
                                                txtButton
                                                txtDisp
                                                txtEdit
                                        vpane
                                project
                                rubber
                                slider
                                stack
                                tray
</EXAMPLE>

<P>
The cosmic class defines the minimal object: a private slot
that lets the object know what class it belongs to; and essential
methods such as create(), destroy(), save(), etc. From here on
the slots and methods definition is rather arbitrary and depends
on what the application is.
<P>
As Viola was designed for visually interactive applications,
most of the classes are GUI widgety oriented. The two notable
exceptions are the socket and TTY classes, which are useful
for communicating with other processes.
<P>
The class hierarchy seems deficient (at this point and from
some point of view, it's probably true) compared to the GUIs
provided by toolkits like Motif. But, it's actually not as
deficient as it seems. For the same reason as Tk, Viola does
not require hard coding of, for example, dialog boxes to
achieve the same functionality.
```

```
<P>
Because of the interpretive nature of the system, complex
GUIs can be composed out of primitive elements, dynamically.
To build a dialog box, a script could be written to create
and necessary objects, and somehow combine them together to
constitute a dialog box.
<P>
As in Tk, making a dialog box can be made easy by calling a
pre written procedure. The current way to do this in Viola is
to build a ``dialog box maker object'', and to send to it a
``Please make me a dialog box, with the following specifications''.
<P>
It's worthwhile to illustrate with an example, which will
show many other aspects of Viola.

</SECTION>

<SECTION NAME="hello.v">
<H2>hello.v</H2>
<EXAMPLE>
\class {txtButton}
\name {hello}
\label {Hello, world!}
\script {
        switch (arg[0]) {
        case "buttonRelease":
                res.dialog("show",
                           "Are you sure you want to exit?",
                           "Yes",       "callback_exit",
                           "No",        "callback_nevermind");
        break;
        case "callback_exit":
                exit(0);
        case "callback_nevermind":
                return; /* do nothing */
        }
        usual();
}
</EXAMPLE>

</SECTION>
</SECTION>

</HMML>
```

**How to Contact the Examiner:**

Any inquiry concerning this communication or earlier communications from the examiner should be directed to St. John Courtenay III, whose telephone number is 571-272-3761. A voice mail service is also available at this number. The Examiner can normally be reached on Monday - Friday, 9:00 AM - 5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the Examiner's Supervisor, Mark Reinhart who can be reached at 571-272-1611.

The fax phone number for the organization where this application or proceeding is assigned is:

**CENTRAL REEXAMINATION UNIT FAX NUMBER:**

**571-273-9900**

**All responses sent by U.S. Mail should be mailed to:**

Commissioner for Patents
PO Box 1450
Mail Stop ex parte REEXAM
Alexandria, VA 22313-1450

Conferee:
M. Reinhart SPRE CRU 3992

[illegible]

ST. JOHN COURTENAY III
PRIMARY EXAMINER

# 831 PH Ex. 21

| | | |
|---|---|---|
| ***Ex Parte* Reexamination Interview Summary** | Control No.<br>90/007,831 | Patent Under Reexamination<br>5,838,906 |
| | Examiner<br>St. John Courtenay III | Art Unit<br>3992 |

All participants (USPTO personnel, patent owner, patent owner's representative):

(1) St. John Courtenay III (3) Michael D. Doyle

(2) Mark Reinhart (4) Charles Krueger

Date of Interview: 18 August 2005

Type: a)☐ Telephonic b)☐ Video Conference
c)☒ Personal (copy given to: 1)☐ patent owner 2)☒ patent owner's representative)

Exhibit shown or demonstration conducted: d)☒ Yes e)☐ No.
If Yes, brief description: Powerpoint presentation of Patent Owner's arguments.

Agreement with respect to the claims f)☐ was reached. g)☐ was not reached. h)☒ N/A.
Any other agreement(s) are set forth below under "Description of the general nature of what was agreed to..."

Claim(s) discussed: 1 and 6.

Identification of prior art discussed: Mosaic (APA), Berners-Lee, Raggett I & II and Toye.

Description of the general nature of what was agreed to if an agreement was reached, or any other comments:
The Patent Owner presented a Powerpoint presentation summarizing the Patent Owner's arguments of record. The Examiner informed the patent owner that OPLA was reviewing the Viola code to determine if it should be considered as a prior art publication.

(A fuller description, if necessary, and a copy of the amendments which the examiner agreed would render the claims patentable, if available, must be attached. Also, where no copy of the amendments that would render the claims patentable is available, a summary thereof must be attached.)

A FORMAL WRITTEN RESPONSE TO THE LAST OFFICE ACTION MUST INCLUDE PATENT OWNER'S STATEMENT OF THE SUBSTANCE OF THE INTERVIEW. (See MPEP § 2281). IF A RESPONSE TO THE LAST OFFICE ACTION HAS ALREADY BEEN FILED, THEN PATENT OWNER IS GIVEN **ONE MONTH** FROM THIS INTERVIEW DATE TO PROVIDE THE MANDATORY STATEMENT OF THE SUBSTANCE OF THE INTERVIEW (37 CFR 1.560(b)). THE REQUIREMENT FOR PATENT OWNER'S STATEMENT CAN NOT BE WAIVED. EXTENSIONS **OF TIME ARE GOVERNED BY 37 CFR 1.550(c).**

ST. JOHN COURTENAY III
PRIMARY EXAMINER

[illegible]

Examiner's signature, if required

cc: Requester (if third party requester)