

EXHIBIT A

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.



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)).

**Notice of Intent to Issue
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
PRIMARY EXAMINER

St. John Courtenay III
Primary Examiner
Art Unit: 3992

cc: Requester (if third party requester)

U.S. Patent and Trademark Office
PTOL-469 (Rev.9-04)

Notice of Intent to Issue Ex Parte Reexamination Certificate

Part of Paper No 20050823

PH_001_0000785907

Ex Parte Reexamination Interview Summary

Control No.	Patent Under Reexamination	
90/006,831	5838906	
Examiner	Art Unit	
St. John Courtenay III	3992	

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

- (1) St. John Courtenay III
- (2) Mark Reinhart
- (3) Michael D. Doyle
- (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.

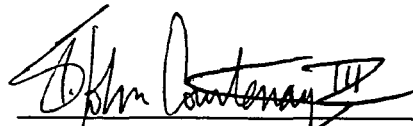
Identification of prior art discussed: Mosaic (APA), Berners-Lee, Raggett I & II, and Tove.

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**



Examiner's signature, if required

cc: Requester (if third party requester)

REEXAMINATION

REASONS FOR PATENTABILITY / CONFIRMATION

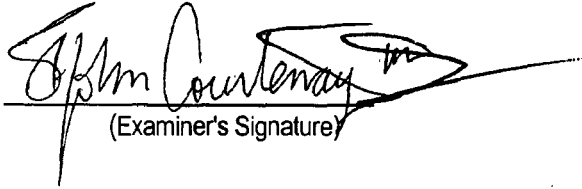
Reexamination Control No. 90/006,831

Attachment to Paper No. 20050823.

Art Unit 3992.

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

ST. JOHN COURTENAY III
PRIMARY EXAMINER



(Examiner's Signature)

**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 image window 352 is within Mosaic window 350 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 In re Cortright, 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 operative coupling 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 parsed 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 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 by their original displaying tools" [MediaMosaic, p. 138, 1st paragraph under "5 Duplicated and Full Views"].

It is reasonable to assume that Toye uses the MediaMosaic "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. MediaMosaic simply redirects a portion of the application display output (containing the object to be embedded) to a "virtual screen" associated with a mapped "Viewport."

MediaMosaic 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 by direct manipulation [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: "users can switch media modes by selecting '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 that must be selected by a mouse to launch an editor application in a separate window" [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 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, 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 selection, and not by parsing. Toye teaches that notebook data is displayed as a data object or filename that must be selected by a mouse to launch an appropriate application in a separate window" [see Toye page 40, 2nd column, paragraph 2; see also page 36, 2nd column, last paragraph, i.e., "... ability to construct hyper-documents containing bitmaps, 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

90/006,831

Art Unit: 3992

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 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).

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) can only be initiated by a manual selection performed by the user (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 can only be invoked manual selection of a data object or file by a user 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 entirety, i.e., including those portions that would argue against obviousness. Panduit Corp. v. Dennison Manufacturing Company, 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 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, etc. In re Mageli, 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); In re Noznick, 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 heavy promotion or advertising 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

90/006,831

Art Unit: 3992

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 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.

**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

90/006,831

Art Unit: 3992

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

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

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 consistent with the specification, where such interpretation is also consistent with the interpretation that those skilled in the art would reach. In re Hyatt, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000); In re Cortright, 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

90/006,831

Art Unit: 3992

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 interpreted operating environment primarily designed for rapid prototyping.

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

90/006,831

Art Unit: 3992

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**

90/006,831

Art Unit: 3992

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



ML 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>
```

90/006,831

Art Unit: 3992

<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 *Hello World* 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 `switch()` 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!}
```


90/006,831

Art Unit: 3992

```
\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 *methods*. 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 *byte codes* 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

90/006,831

Art Unit: 3992

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.

90/006,831

Art Unit: 3992

<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 designed 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.

90/006,831

Art Unit: 3992

<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 versatility 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 subclassing 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>

90/006,831

Art Unit: 3992

<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.

90/006,831

Art Unit: 3992

<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>

90/006,831

Art Unit: 3992

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



ST. JOHN COURTENAY III
PRIMARY EXAMINER