UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

DISNEY ENTERTAINMENT & SPORTS LLC,
Petitioner

v.

ADEIA GUIDES INC.,
Patent Owner.

Case No. IPR2026-00053
U.S. Patent No. 9,235,428

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,235,428

TABLE OF CONTENTS

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

TABLE OF CONTENTS
(continued)

Page

LIST OF EXHIBITS[1]

Ex. No.	Description
Ex-1001	U.S. Patent No. 9,235,428 ("the '428 Patent")
Ex-1002	Declaration of Dr. Brian Anthony
Ex-1003	Curriculum Vitae of Dr. Brian Anthony
Ex-1004	Prosecution History of U.S. Patent No. 9,235,428
Ex-1005	U.S. Patent No. 6,886,013 to Beranek ("Beranek")
Ex-1006	U.S. Patent No. 6,188,401 to Pleyer ("Pleyer")
Ex-1007	U.S. Patent No. 6,208,336 to Carter ("Carter")
Ex-1008	U.S. Patent No. 6,473,407 to Ditmer ("Ditmer")
Ex-1009	M. Satyanarayanan, Jason Flinn & Kevin R. Walker, *Visual Proxy: Exploiting OS Customizations without Application Source Code* (Carnegie Mellon Univ., Sch. of Comput. Sci., July 1, 1999)
Ex-1010	Badriprasad Keshavamurthy, *Software Architecture and Cross Platform User Interfaces* (Master's thesis, Tex. Tech Univ., Aug. 1996)
Ex-1011	Thomas Tilley, *Distributing a Platform Independent Graphical User Interface* (1998)
Ex-1012	David S. Wishart et al., *A platform-independent graphical user interface for SEQSEE and XALIGN*, 13 Comput. Appl. Biosci. 561 (1997)

[1] Four-digit pin citations that begin with 0 are to the branded numbers added by Disney in the bottom right corner of the exhibits. All other pin citations are to original page, column, paragraph, or line numbers.

Ex. No.	Description
Ex-1013	Frank Kargl, Torsten Illmann, Michael Weber & Stefan Ribhegge, *Dynamic User Interfaces with Java* (Distributed Systems Dep't, Univ. of Ulm, n.d.)
Ex-1014	INTENTIONALLY LEFT BLANK
Ex-1015	INTENTIONALLY LEFT BLANK
Ex-1016	INTENTIONALLY LEFT BLANK
Ex-1017	INTENTIONALLY LEFT BLANK
Ex-1018	INTENTIONALLY LEFT BLANK
Ex-1019	INTENTIONALLY LEFT BLANK
Ex-1020	Claim Mapping Table
Ex-1021	*Adeia Techs., Inc. v. The Walt Disney Co.*, No. 1:24-cv-01231, Defendants' Supplemental Disclosure of Proposed Claim Constructions (D. Del. Sept. 16, 2025)
Ex-1022	*Adeia Techs., Inc. v. The Walt Disney Co.*, No. 1:24-cv-01231, Plaintiffs' Initial Proposed Claim Constructions (D. Del. Sept. 5, 2025)

I. INTRODUCTION

Disney Entertainment & Sports LLC ("Petitioner") requests *inter partes* review ("IPR") of Claims 1-5 and 9-13 of U.S. Patent No. 9,235,428 ("the '428 Patent") (Ex-1001), currently assigned to Adeia Guides Inc. ("PO").

II. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8

Real Parties-in-Interest: Petitioner identifies the following real parties-in interest: Disney Entertainment & Sports LLC, The Walt Disney Company, Disney Media and Entertainment Distribution LLC, Disney DTC LLC, Disney Streaming Services LLC, Disney Platform Distribution, Inc., Bamtech, LLC, Hulu, LLC, and ESPN, Inc.

Related Matters: PO has asserted the '428 Patent against Petitioner in *Adeia Technologies, Inc., et al. v. The Walt Disney Company, et al.*, No. 1:24-cv-01231-MN (D. Del.) ("Disney litigation").

Lead and Backup Counsel:

- Lead Counsel:

Benjamin M. Haber (Reg. No. 67,129)
O'Melveny & Myers LLP
400 South Hope Street, 18th Floor
Los Angeles, CA 90071
Telephone: (213) 430-6000
Fax: (213) 430-6407
Email: bhaber@omm.com

- Backup Counsel:

William M. Fink (Reg. No. 72,332)
O'Melveny & Myers LLP
1625 Eye Street, NW
Washington, DC 20006
Telephone: (202) 383-5300
Fax: (202) 383-5414
Email: tfink@omm.com

Xin-Yi Zhou (Reg. No. 63,366)
O'Melveny & Myers LLP
400 South Hope Street, 18th Floor
Los Angeles, CA 90071
Telephone: (213) 430-6000
Fax: (213) 430-6407
Email: vzhou@omm.com

Caitlin P. Hogan (Reg. No. 61,515)
O'Melveny & Myers LLP
1301 Avenue of the Americas, Suite 1700
New York, NY 10019
Telephone: (212) 326-2000
Fax: (212) 326-2061
Email: chogan@omm.com

Service Information: Petitioner consents to electronic service by email to the following addresses:

- bhaber@omm.com
- tfink@omm.com
- vzhou@omm.com
- chogan@omm.com
- ommdisneyadeia@omm.com

III. FEE AUTHORIZATION

The PTO is authorized to charge any fees due during this proceeding to Deposit Account No. LA500639.

IV. GROUNDS FOR STANDING

Petitioner certifies that the '428 Patent is available for review, and Petitioner is not barred or estopped from requesting review.

V. PRECISE RELIEF REQUESTED

Petitioner requests review and cancellation of Claims 1-5 and 9-13 as unpatentable based on the following grounds, supported by a declaration from Dr. Brian W. Anthony. Ex-1002 ¶¶1-32, 44-181; Ex-1003.

Ground	Summary
1	Claims 1-5 and 9-13 are obvious over Beranek (Ex-1005) in view of Pleyer (Ex-1006)
2	Claims 3 and 11 are obvious over Beranek-Pleyer in view of Carter (Ex-1007)
3	Claims 9-11 are obvious over Ditmer (Ex-1008) in view of Beranek

VI. THE CHALLENGED PATENT

The '428 Patent relates to a computer implemented system that provides a "graphical user interface (GUI) for an application program" using a "renderer, a proxy and a workflow manager." Ex-1001, Abstract.

As shown below in Figure 7, renderer 200 (green) communicates with an application proxy 205 (purple) and receives "instantiated user interface components 208" and "instantiated document viewer 212" (blue) to render the graphical user interface (GUI). Ex-1001, 4:7-10, 5:17-18. User interface components are "controls such as button, sliders, radio buttons, edit boxes, etc.," that can be implemented using "ActiveX controls, Java applets, etc." *Id.*, 5:30-34. Document viewers include "viewers appropriate to observe and interact with data appropriate to the application program, in all of its possible contents." *Id.*, 5:34-37.

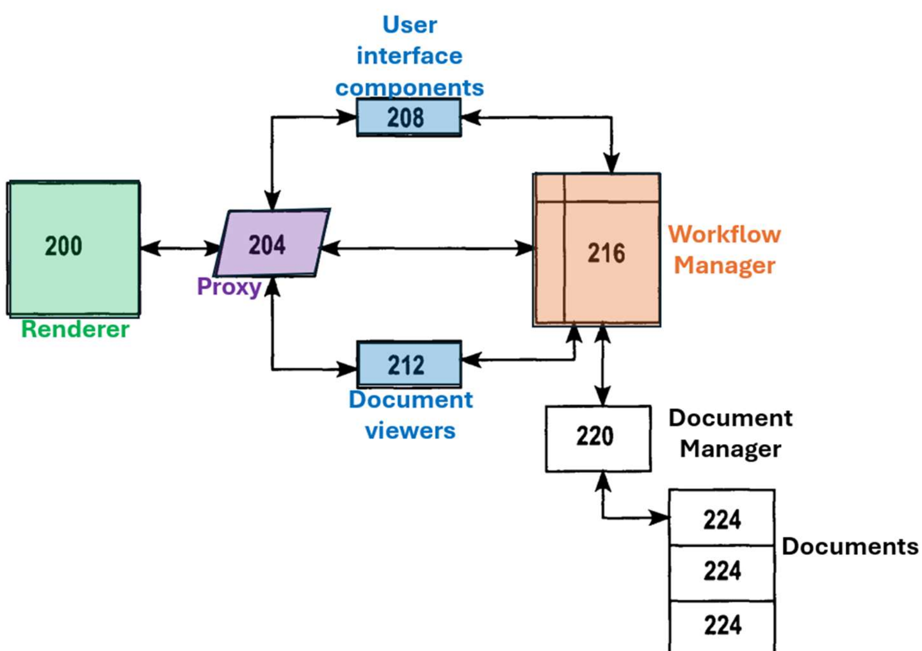


FIG. 7

Proxy 204 appears "as a single, monolithic application program and all details of the components which make up the application program are essentially hidden from renderer 200 which need only interface to application proxy 204." Ex-1001, 5:8-13. In response to a "context change, or other event requiring a display update occurs within the application program, either as a result of a user interface event or another event," proxy 204 determines the changes required to be made to the application program user interface, including rendering a new layout or updating the present layout. *Id*., 6:55-66. The proxy sends a message to the renderer 200, which will then re-render the application program user interface to dynamically update the display to reflect the changes. *Id*. Application proxy 204 also communicates to a "workflow manager 216" (orange) for the updated elements to be rendered. *Id*., 4:30-34, 6:66-7:5; Ex-1002 ¶¶33-36.

VII. PROSECUTION HISTORY OF THE '428 PATENT

The '428 Patent was filed August 23, 2006, as a continuation of Application No. 09/240,844, filed on February 1, 1999, now Pat. No. 7,117,446.

The '428 Patent was allowed only after a single round of examination and an appeal to the Board. The Examiner rejected the claims under 35 U.S.C. §§ 101, 103. In response, the Applicant argued that U.S. Patent No. 6,038,395 to Chow et al. ("Chow") did not teach a "single monolithic application program" because Chow's proxy manipulates and edits the proxy objects "at build time," and that the claims require the proxy "to determine changes required to be made to the GUI for the application program during execution of the application program." Ex-1004, 83. The Examiner responded that Chow "discloses that Interface repository being contained in the proxy object itself therefore is a monolithic application and there could only one proxy object," and again rejected the claims as being obvious. The Examiner further added U.S. Patent No. 7,168,086 to Carpenter to the Lynch-Chow combination for its teaching of a proxy server that determines the user interface elements to be presented on the client terminal's viewer. *Id*., 93-94.

The Applicant subsequently appealed the Examiner's rejection of the recited proxy and workflow manager to the Board. Specifically, Applicant argued that Chow's proxy is not a "single monolithic application program" because the proxy would have been understood as using "multiple proxy objects, where each proxy object corresponds to one target object." Ex-1004, 127-130. Applicant also argued that Lynch did not teach the "workflow manager" because it merely teaches a "formatter 707, which "insert[s] proper formats, such as line breaks," and Applicant could not discern how the surface formatting suggests the recited workflow manager that communicates document viewers and user interface components. Ex-1004, 130.

On May 5, 2015, the Board reversed the Examiner's rejection, finding that the Examiner had not adequately identified a "workflow manager" in the primary reference, Lynch. Ex-1004, 189-190. The '428 Patent was allowed based on Examiner's amendment to overcome the pending § 101 rejection. Ex-1004, 200; *see also id*., 248 (granting after Applicant's minor amendment).

During prosecution, the Examiner listed nineteen references on its Notice of References cited forms. *See* Ex-1004, 58-59, 101-102. U.S. Patent No. 6,473,407 to Ditmer et al. was listed but never applied to the claims. *Id*., 58, 101. The Examiner materially erred by not considering the Ditmer during prosecution because, as demonstrated in Ground 3 below, Ditmer, in view of Beranek, respectively, renders obvious each limitation of Claims 9-11. Ex-1002 ¶¶37-43. It was a material error to not apply Ditmer's teachings to the amended claims during prosecution.

Besides Ditmer, this Petition presents new arguments and prior art not considered during prosecution, and nothing precludes considering them in the first instance under 35 U.S.C. § 325(d).

VIII. LEVEL OF ORDINARY SKILL IN THE ART

A person of ordinary skill in the art at the relevant time ("POSITA") would have had a bachelor's degree in electrical engineering, computer science, computer engineering, or a related field, and two-to-three years of experience in the research, design, development, or testing of interactive entertainment streaming interface and display technologies, user interfaces, and platforms, with additional education substituting for experience and vice versa. Ex-1002 ¶¶24-28.

IX. PRIORITY DATE

For purposes of this proceeding only, Petitioner applies the earliest alleged priority date of the ’428 Patent: Feb. 1, 1999.[2] Accordingly, pre-AIA 35 U.S.C. § 102 applies and the ’428 Patent was never eligible for PGR.

X. CLAIM CONSTRUCTION

For the purposes of this Petition and pursuant to 37 C.F.R. § 42.104(b)(3), Petitioner’s proposed constructions for these terms are below. Ex-1002 ¶¶72-84. Petitioner does not currently believe that any other terms require express construction. Consolidated Trial Practice Guide, 44, 45.

A. “a renderer to render a graphical user interface (GUI) for an application program” (Claim 9)

In the related litigation, Petitioner proposed that this term is a means-plus-function elements under 35 U.S.C. § 112 ¶6:

Function: “to render a graphical user interface (GUI) for an application program”

Structure: Insufficient structure; indefinite.

PO has proposed that this limitation should be given a plain and ordinary meaning. Ex-1022. For the purposes of this proceeding, Petitioner has applied this limitation to include software that renders a graphical user interface for an

[2] Petitioner respectfully reserves the right to challenge the priority date in other proceedings.

application program in the same way as described by the patent, such as with "a DHTML browser." Ex-1001, 4:53, 5:23-24, 7:19; Ex-1002 ¶¶74-75.

B. "single monolithic application program" (Claim 9)

In the related litigation, Petitioner proposed that the "single monolithic application program" recited in independent Claim 9 means a "single self-contained application program." Ex-1021. PO proposes that "the proxy appears to the renderer as a single monolithic application program" means "details of the components that make up the application program are hidden from the renderer." Ex-1022. For the purposes of this proceeding, Petitioner has applied this limitation to include software that operates in the same way as "application proxy 204," which is explained to be "a single, monolithic application program and all details of the components which make up the application program are essentially hidden from renderer 200 which need only interface to application proxy 204." Ex-1001, 5:8-12; Ex-1002 ¶¶76-77.

C. "the ox" (Claim 9)

In the related litigation, Petitioner proposed that the recited "ox" is not amendable for construction. *See* Ex-1021. PO has proposed that "the ox" is "the proxy." *See* Ex-1022. For the purposes of this proceeding, Petitioner applies this term as if the incomprehensible error includes "the proxy." Ex-1002 ¶78.

D. "a workflow manager to communicate one or more document viewers and associated user interface components to the proxy" (Claim 9)

In the related litigation, Petitioner proposed that this term is a means-plus-function element under 35 U.S.C. § 112 ¶6:

Function: "to communicate one or more document viewers and associated user interface components to the proxy"

Structure: Insufficient structure; indefinite. Ex-1021.

PO has proposed that this limitation should be given a plain and ordinary meaning. Ex-1022. For the purposes of this proceeding, Petitioner has applied this limitation to include software that operates in the same way as "workflow manager 216," which "manages communication between document viewers 212, user interface components 208 and, through application proxy 204, with renderer 200." Ex-1001, 5:57-59, Fig. 2; Ex-1002 ¶¶79-80.

E. "wherein the proxy functions to hide details of each component which makes up the application program from the renderer" (Claim 10)

In the related litigation, Petitioner proposed that this limitation is indefinite. *See* Ex-1021. PO has proposed that the limitation is not indefinite. *See* Ex-1022. For the purposes of this proceeding, Petitioner has applied this limitation to include that the proxy's component functionality "is not visible to the user." Ex-1001, 5:23-25; Ex-1002 ¶81.

F. "wherein the proxy functions to enable each component which makes up the application program to be modified in a manner that is transparent to the renderer" (Claim 11)

In the related litigation, Petitioner proposed that this limitation is indefinite. *See* Ex-1021. PO has proposed that the limitation is not indefinite but did not provide a proposed construction. *See* Ex-1022. During a meet-and-confer in the related litigation, PO stated that it interpreted "transparent to the renderer" as requiring that the renderer is not aware of any modification of "each component which makes up the application program." Another possible interpretation of this limitation is that the proxy's component functionality is "transparent to the renderer" such that the renderer can see through the proxy to additional control options that may be available. Ex-1001, 5:12-14. For the purposes of this proceeding, Petitioner has applied this limitation in both possible alternatives to address either of the interpretations outlined above. Ex-1002 ¶¶82-84.

XI. BRIEF DESCRIPTION OF THE APPLIED PRIOR ART

A. Beranek (Ex-1005)

Beranek (U.S. Patent No. 6,886,013) was filed on August 20, 1997, issued on April 27, 2004, and is prior art to the '428 Patent under at least § 102(e). Ex-1002 ¶¶60-62.

Beranek teaches "[a] method of controlling presentation on a client of a Web document formatted according to a markup language and supported on a server, the client including a browser and connectable to the server via a computer network." Ex-1005, 15:33-36 (Cl. 1). As shown in Figure 9 below, a browser "sits on a given machine (e.g., a personal computer having its own display system)". Ex-1005, 13:2-6. Beranek teaches that a "reformatted Web document is then passed to the browser for display." Ex-1005, 3:7-8, Abstract.

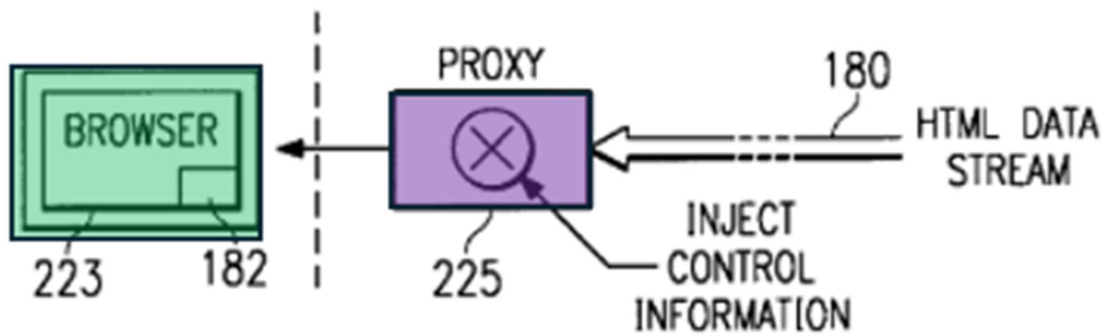


FIG. 9

Beranek teaches a "client-side caching proxy," which "intercept[s] the Web document and then dynamically rewrite[s] the document before it is displayed on the browser." Ex-1005, 2:63-3:2. This enables the "proxy to dynamically adjust the 'look and feel' of a given Web document at the client side of the connection." *Id*., 8:39-42. The proxy "is implemented in software and includes a cache 227." *Id*., 8:24-26. Specifically, proxy 225 (purple), intercepts the data stream and "inject[s] new control information … in order to affect how the Web content is ultimately displayed." *Id*., 13:9-14. The proxy is used to modify and/or process data before it is displayed on the user's interface, such that the "data font or presentation to be modified, supplemented or otherwise adapted to the particular display environment." *Id*., 12:17-19. Beranek's proxy thus provides a tool to enhance and tailor the "look and feel" of data displayed and enables the "user interface to be tailored to the Web appliance environment." Ex-1005, 12:19-23.

B. Pleyer (Ex-1006)

Pleyer (U.S. Patent No. 6,188,401) was filed on March 25, 1998, and issued on February 13, 2001, and is prior art to the '428 Patent under at least § 102(e). Ex-1002 ¶¶63-65.

Pleyer teaches a script-based user interface implementation defining components using a text markup language. Ex-1006, Title. Pleyer teaches a browser program, as shown in Figure 4 below, which includes a user interface component 150 that renders a variable message, scroll indicators, and a connection status area, an internet component 152 that renders internet web content, a television component 154 that renders a live video stream. Ex-1006, 6:40-46, 6:50-51, 6:60-65. Such user interface controls and dialogs are "implemented as HTML files or resources" with the overall "logic and sequence of the user interface" being "established by a global script file written in a scripting language such as JavaScript." *Id*., 7:18-23.

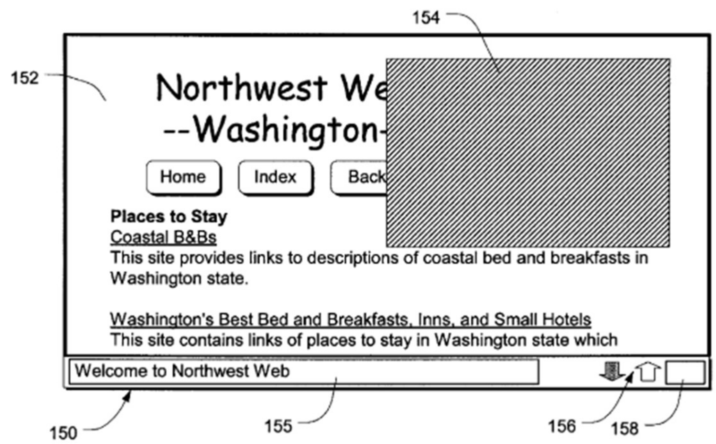


Fig. 4

As shown below in Figure 3, Pleyer teaches a supervisory application 100 that "communicates with hardware devices such as the network interface television tuner, keyboard, and display components though conventional Windows operating system calls." Ex-1006, 5:55-65. The supervisory application communicates the global script 106, which includes browsable content 104 and UI components 105. Pleyer teaches that "controls and dialogs … are implemented as text and graphics documents written in … HTML," and that in order to present options to a user, "global script 106 generates dialog boxes and controls by specifying local HTML graphical elements." *Id.*, 7:40-45. JavaScript is used to "add interactivity to HTML, to allow for user interaction and feedback, multimedia and animation, and to link HTML to other technologies such as Java, Active X, and plug-ins." *Id.*, 4:12-16. For example, Pleyer teaches that "event handlers implemented by global script 106" includes generating an "on-screen keyboard" and "audio rendering" (*id.*, 9:1-3, 9:29-

30), and "create[ing] a new user interface component in response to a user pressing a specific key." *Id.*, 9:49-51.

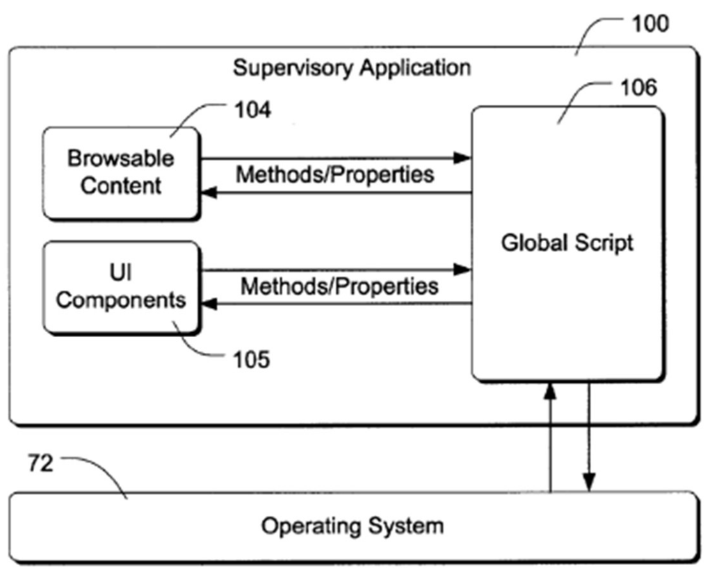


Fig. 3

C. Carter (Ex-1007)

Carter (U.S. Patent No. 6,208,336) was filed on March 20, 1998, and issued on March 27, 2001, and is prior art to the '428 Patent under at least § 102(e). Ex-1002 ¶¶66-68.

Carter teaches a "dynamic graphical user interface feature-set configuration." Ex-1007, Title. Carter teaches determining which command classes to "instantiate" and constructing "the graphical user interface for the application such that the graphical user interface contains the additional functions of the successfully instantiated command classes." Ex-1007, Abstract. Carter teaches a constructor 3120 which "locates any packages (or classes in a package)." Ex-1007, 5:27-37. The program thus determines "which classes are available and then construct[s] a graphical user interface ***to reflect all of the classes***." *Id*., 2:56-57. Figure 3 shows a command class library which stores multiple command classes 3100-3400, which contain additional functionality of an application. *Id*., 4:27-30.

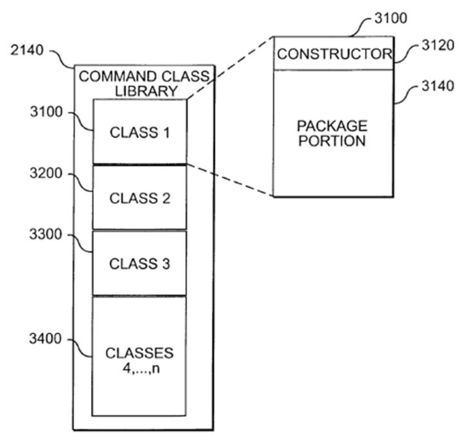


FIG. 3

Carter's command classes are rendered on the application so that the user may select the additional functionality, which then causes a new layout in the GUI and updating a present layout in the GUI for the user. *See, e.g.*, Figs. 5A, 5B. In Figure 5B, a GUI presents four basic controls of "In Box 5110, Out Box 5120, Trash 5130, and Send Mail 5140" and additional controls for a mail application, by instantiating one or more command classes, such as "a Spell Checker 5250 and a Thesaurus 5260." *Id*., 5:47-56; Ex-1002 ¶68.

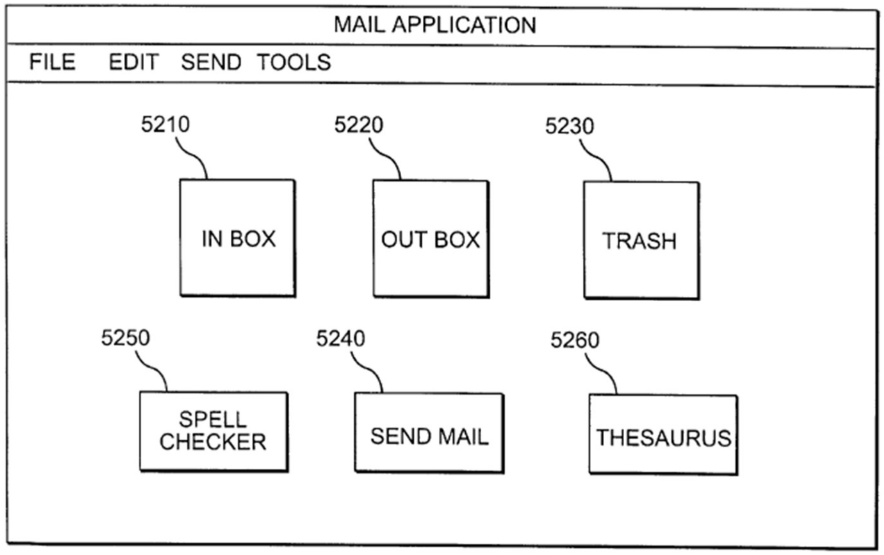


FIG. 5B

D. Ditmer (Ex-1008)

Ditmer (U.S. Patent No. 6,473,407) was filed on September 24, 1998, and issued on October 29, 2002, and is prior art to the '428 Patent under at least § 102(e). Ex-1002 ¶¶69-71.

Ditmer teaches a "Web-based fault and alarm management tool that enables customers to monitor and analyze the performance of their voice and data network via a graphical user interface." Ex-1008, Abstract; *see also id.*, 3:27-36. Ditmer provides a "system and a method for communicating information relating to the dedicated voice and data networks, from an enterprise to a customer at a client workstation, the present invention includes a client browser application located at the client workstation." *Id.*, 3:52-56. Ditmer states that customer browser is enabled and includes "***client applications responsible for presentation*** and front-end services." *Id*., 6:14-16

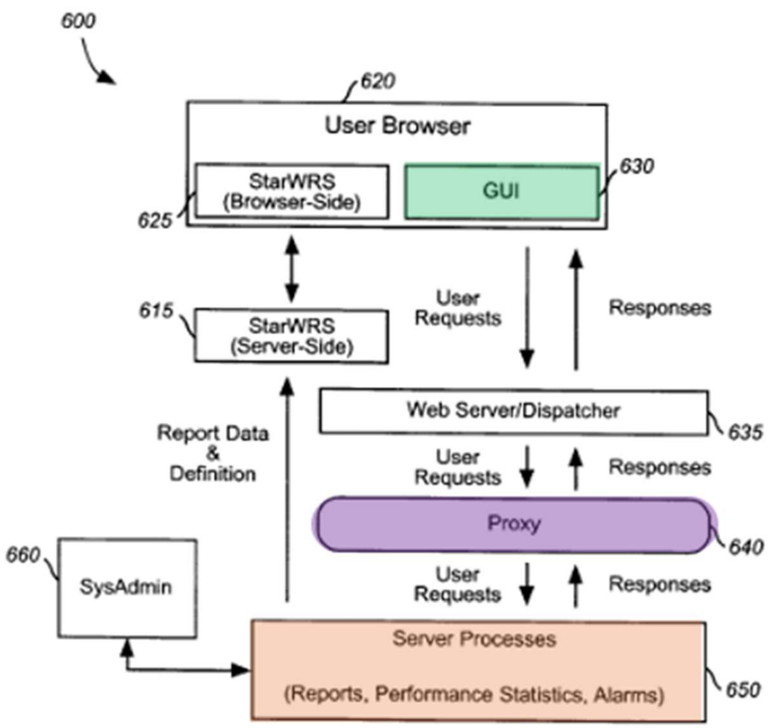


FIG. 6

Ditmer teaches an "event monitor system," which includes a three-tier architecture. As shown in Figure 6, first, GUI 630 (green) allows a user to submit requests and receive responses. Second, Ditmer teaches that a user's request is forwarded to "an integrated proxy application 640 [(purple)] for receiving and interpreting the user messages and enabling the event monitor functionality." Ex-1008, 14:31-34. Specifically, the proxy examines and confirms incoming requests to ensure that "they include validly formatted messages for the service with acceptable parameters; a translation process for translating a message into the database query message or networking protocol; and, a management process for managing the communication of the specific customer request with the middle-tier server to actually get the request serviced." *Id.*, 23:7-14; *see also id.*, 24:14-20. The "[d]ata returned from the … server is translated back to client format, if necessary, and returned to the dispatch server as a response to the request." *Id.*, 23:14-17; *see also id.*, 9:46-50. While shown as a separate module, the proxy "may reside on the dispatch server 635 itself." Ex-1008, 24:2-4. Third, Ditmer's "event monitor server 650 [(orange)] is responsible for all tasks leading up to and including the management of alarms and performance reports including data collection, calculation, storage, and report generation." Ex-1008, 14:58-65.

XII. DETAILED EXPLANATION OF THE UNPATENTABILITY GROUNDS

A. Ground 1: Claims 1-5 and 9-13 are obvious over Beranek in view of Pleyer

1. A POSITA would have been motivated to combine Beranek's teachings with Pleyer's teachings and would have had a reasonable expectation of success.

A POSITA would have been motivated to combine the teachings of Beranek and Pleyer and had a reasonable expectation of success in doing so because each relates to controlling how a document is presented for display on a browser of a client machine. Ex-1005, Title, Abstract; Ex-1006, Title, Abstract; Ex-1002 ¶¶85-98.

Beranek teaches "[a] method of controlling presentation on a client of a Web document formatted according to a markup language and supported on a server, the client including a browser and connectable to the server via a computer network." Ex-1005, 15:33-36 (Cl. 1). Beranek's proxy intercepts, reformats, and controls how a Web document appears in the browser, to the user. Ex-1005, 2:54-61, 2:63-3:2. The proxy will "re-format the Web document according to some given protocol or filter property," such as "adjust[ing] or modify[ing] foreground/background colors, overrid[ing] font types and size, control[ing] display geometry (e.g., by reserving some dedicated display area), chang[ing] URLs or display[ing] other URLs, embed files, add, remove or reconfigure frames, or control text and table formats." *Id.*, 3:5-14. This enables the "proxy to dynamically adjust the 'look and feel' of a given Web document at the client side." *Id.*, 8:39-42. The "reformatted Web document is then passed to the browser for display." Ex-1005, 3:7-8, Abstract. Beranek teaches that, as the Web document is received from the server, "the HTML is parsed to identify the format of the document and the information therein." Ex-1005, 3:2-5.

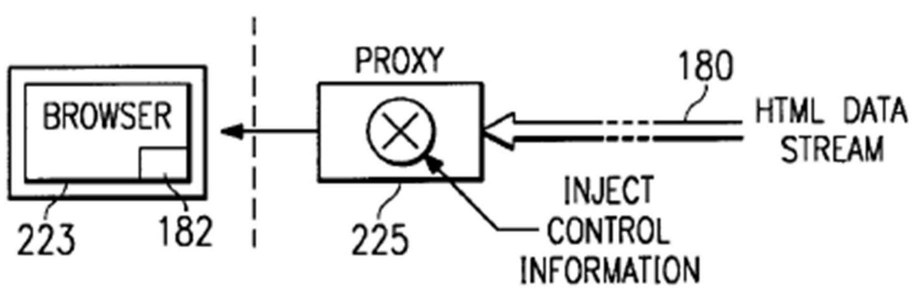


FIG. 9

Pleyer teaches a supervisory application 100 that "communicates with hardware devices such as the network interface television tuner, keyboard, and display components though conventional Windows operating system calls." Ex-1006, 5:55-65. The supervisory application communicates the global script 106, which includes browsable content 104 and UI components 105. Pleyer teaches that "controls and dialogs… are implemented as text and graphics documents written in … HTML," and that in order to present options to a user, "global script 106 generates dialog boxes and controls by specifying local HTML graphical elements." *Id*., 7:40-45. JavaScript is used to "add interactivity to HTML, to allow for user interaction and feedback, multimedia and animation, and to link HTML to other technologies such as Java, Active X, and plug-ins." *Id*., 4:12-16. For example, Pleyer teaches that "event handlers implemented by global script 106" includes generating an "on-screen keyboard" and "audio rendering" (*id*., 9:1-3, 9:29-30), and to "create a new user interface component in response to a user pressing a specific key." *Id*., 9:49-51.

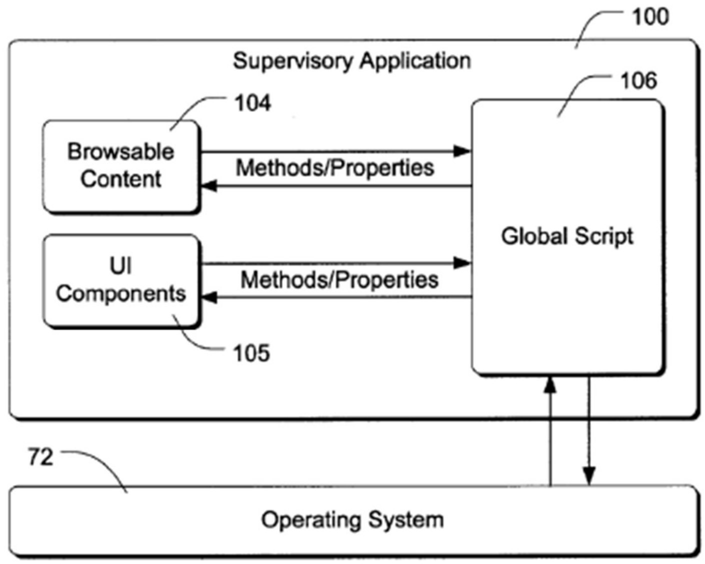


Fig. 3

A POSITA would have been motivated to communicate Beranek's web document from the data stream to the proxy with document viewer(s) and associated user interface components, as taught by Pleyer. Ex-1002 ¶¶86-90. A POSITA would have understood that Beranek's proxy receives a web document from a workflow manager, which includes a script that implements document viewers and associated user interface components to display the requested Web document. A POSITA would have further understood that Pleyer provides an example of this well-known approach. Specifically, Pleyer teaches user interface controls and dialogs are "implemented as HTML files or resources" with the overall "logic and sequence of the user interface" being "established by a global script file written in a scripting language such as JavaScript." Ex-1006, 7:18-23. The global script file comes from the supervisory application, a workflow manager. A POSITA would have been motivated for Beranek's workflow manager to communicate one or more document viewers with associated user interface components, implemented by a script, as taught by Pleyer, to control the interaction of displayed controls with underlying hardware and display appropriate graphic elements. Ex-1006, 10:35-40; Ex-1002 ¶¶89-90. Specifically, Pleyer teaches that its supervisory application program (workflow manager) runs under an operating system and implements a global script (Ex-1006, 2:11-18), which allows "a software manufacturer [to] sell a single product … to many different manufacturers" (*id*., 2:42-43) and provides a "very easy" way for a "customer to create an entirely new user interface, and to adapt it to new hardware, without distributing more critical underlying aspects of the software product." *Id*., 2:49-54. Moreover, Pleyer teaches that its supervisory application program's global script communicates with "HTML components" to produce a full and flexible user interface. *Id*., 2:38-41. A POSITA would have been motivated to implement Pleyer's teachings into Beranek to achieve these additional benefits taught by Pleyer. Ex-1002 ¶¶89-90.

A POSITA would have had a reasonable expectation of success in combining the teachings of Beranek and Pleyer. Ex-1002 ¶¶91-98. Indeed, as shown below in Modified Figure 9' below, Pleyer's supervisory application, which includes its global script with associated browsable content and UI components would have been implemented in conjunction with Beranek's architecture to carry out Beranek's functions (the implementation details of which are left to a POSITA): receiving the content to be passed to the renderer in the appropriate format. *Id*. A POSITA would be motivated to implement Pleyer's teachings with Beranek to "provide[] a very flexible way of implementing a user interface, which allows individual hardware manufacturers to customize the user interface in an infinite number of ways." Ex-1006, 10:58-60.

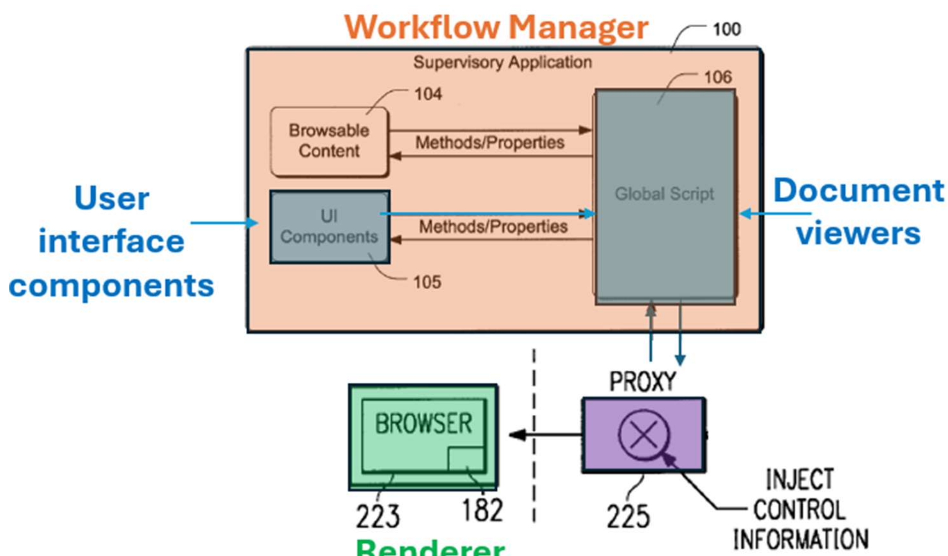


FIG. 9'

Moreover, both references describe methods for presenting a requested web document to a user in an appropriate format on a graphical user interface by the user using similar hardware and software components. Ex-1005, Title, 15:33-36 (Cl. 1); Ex-1006, Title, Abstract. A POSITA would thus have understood that these references disclose interrelated teachings based on well-understood technologies that would have been amenable to various well-understood and predictable combinations. Ex-1002 ¶¶95-98.

2. Independent Claims 1 and 9

a. Element 1[pre]: A computer implemented system for providing a graphical user interface, comprising:

Element 1[A]: a non-transitory computer readable medium;

Element 9[pre]: A method for providing a graphical user interface, comprising:

Beranek-Pleyer teaches the preambles of claims 1 and 9, to the extent that they are limiting. Ex-1002 ¶¶99-101. Beranek teaches "[a] method of controlling presentation on a client of a Web document formatted according to a markup language and supported on a server, the client including a browser and connectable to the server via a computer network." Ex-1005, 15:33-36 (Cl. 1). Beranek also teaches "[a] ***computer program product in computer-readable media*** for use in controlling presentation on a client of a Web document formatted according to a markup language (HTML) and supported on a server, the client including a browser and connectable to the server via a computer network." Ex-1005, 16:43-47 (Cl. 19).

Beranek teaches that the browser "sits on a given machine (e.g., a personal computer having its ***own display system***…"). Ex-1005, 13:2-6. As shown in Figure 9 below, browser 232 is available on a display screen, such as a graphical user interface. Moreover, Beranek teaches a "client computer, for example, a Web appliance, [] connectable to the Internet and comprises a processor having an operating system, a browser application run by the operating system, a cache, and a proxy." Ex-1005, 2:54-58; *see also id.*, 4:58-5:17 ("client machine is a personal computer such as a desktop of notebook computer"). A POSITA would have understood that Beranek's method for controlling presentation on a client computer is a method for providing a graphical user interface because the client computer would have a graphical user interface to display the Web document that Beranek controls and displays. Ex-1002 ¶101.

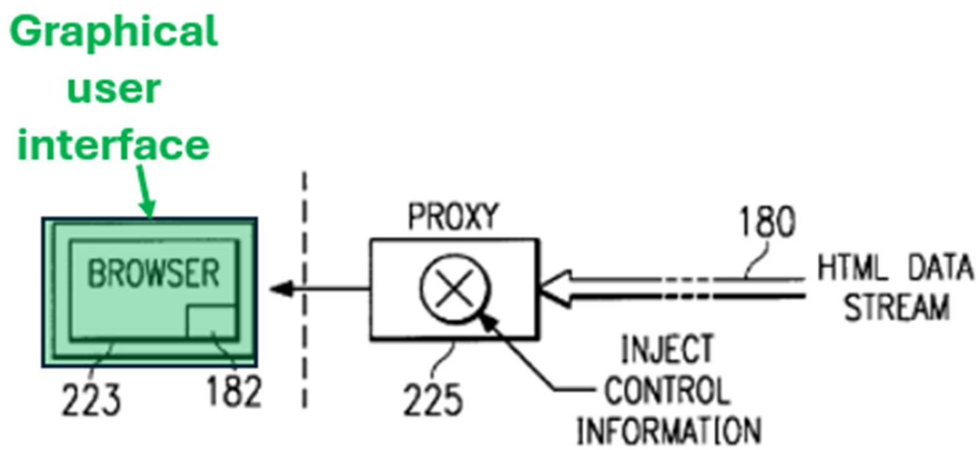


FIG. 9

b. Elements 1[B]/9[A]: [operating] a renderer defined to render a graphical user interface (GUI) for an application program;

Beranek-Pleyer teaches elements 1[B]/9[A]. Ex-1002 ¶¶102-104.

Beranek teaches operating a renderer defined to render a GUI for an application program. As discussed in Section X, for the purposes of this proceeding, Petitioner has treated this limitation as including software that renders a graphical user interface for an application program that same way as the patent, which would include software such as "a DHTML browser." Ex-1001, 4:53, 5:23-24, 7:19.

Beranek's method operates a renderer to render a GUI for the application program by teaching that its method intercepts, reformats, and controls how a Web document ***appears*** in the browser to the user. Ex-1005, 2:54-61. Specifically, Beranek teaches that the "reformatted Web document is then passed to the ***browser for display***." Ex-1005, 3:7-8, Abstract. Beranek further explains that an "object of this invention is to implement a 'dynamic HTML' [DHTML] function in a standalone Web appliance or system to thereby modify the appearance of a web document prior to its display on the browser." *Id*., 2:36-39. A POSITA would have understood Beranek's software instructs the browser application to render a GUI for the application program using its "dynamic HTML" (DHTML) functioning browser. This is consistent with the description in the '428 Patent, stating that its "renderer 200 is a DHTML browser." Ex-1002 ¶¶103-104; Ex-1001, 5:23-34.

c. Elements 1[C]/9[B]: [(i) [operating] a proxy defined to provide each component necessary for rendering the GUI to the renderer, [(ii)] wherein the proxy appears to the renderer as a single monolithic application program, and [(iii)] wherein the proxy is defined to determine changes required to be made to the GUI for the application program during execution of the application program, [(iv)] wherein a given change determined by the [proxy]/[ox] is one of rendering a

new layout in the GUI and updating a present layout in the GUI; and

Beranek-Pleyer teaches elements 1[C]/9[B]. Ex-1002 ¶¶105-110.

Beranek-Pleyer teaches operating a proxy defined to provide each component necessary for rendering the GUI to the renderer. Ex-1002 ¶106. Specifically, Beranek teaches a "client-side caching proxy," (purple below) that "intercept[s] the Web document and then dynamically rewrite[s] the document before it is displayed on the browser." Ex-1005, 2:63-3:2. This enables the "proxy to dynamically adjust the 'look and feel' of a given Web document at the client side of the connection." *Id.*, 8:39-42. The proxy "is implemented in software and includes a cache 227." *Id.*, 8:24-26. Specifically, as shown in Figure 9, proxy 225 (purple), intercepts the data stream and "inject[s] new control information … in order to affect how the Web content is ultimately displayed." *Id.*, 13:9-14.

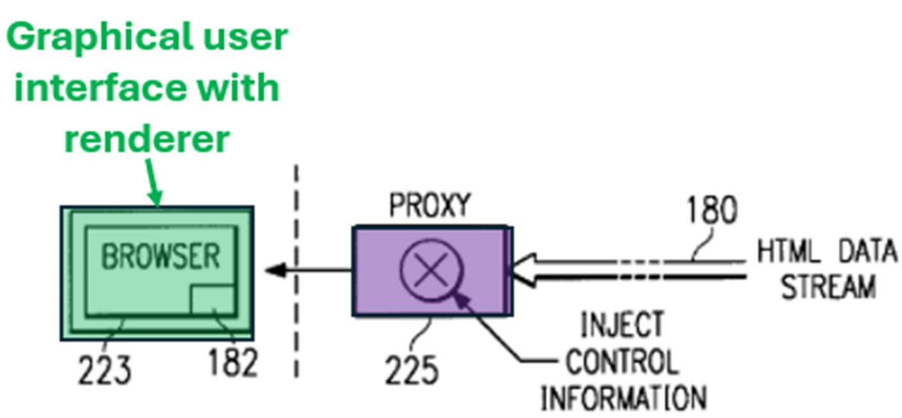
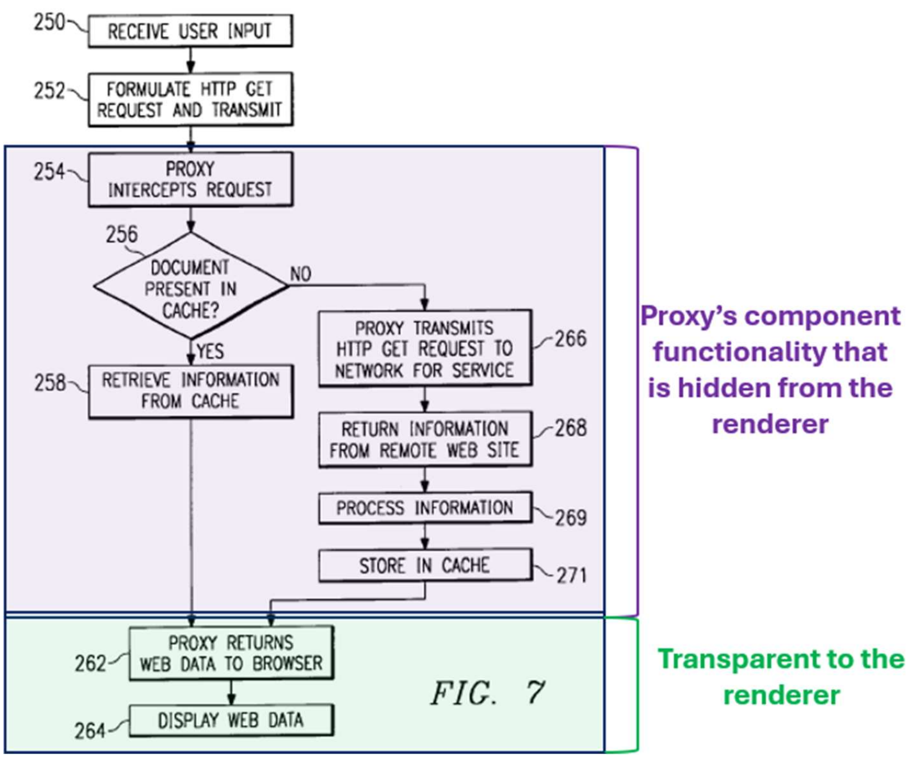


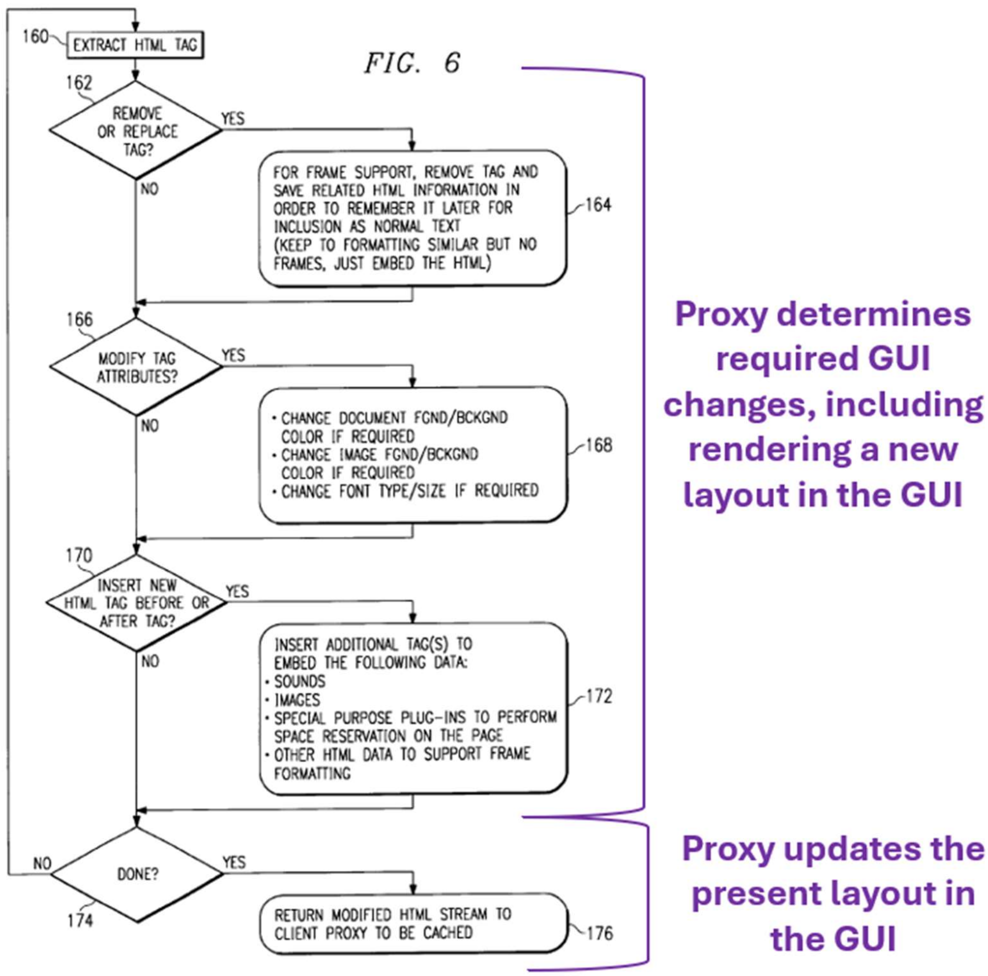
FIG. 9

In Beranek, the proxy appears to the renderer as a "single monolithic application program." Ex-1002 ¶¶107-109. As discussed in Section X, for the purposes of this proceeding, Petitioner applied this limitation to include software that operates in the same way as "application proxy 204," which is "a single, monolithic application program and all details of the components which make up the application program are essentially hidden from renderer 200 which need only interface to application proxy 204." Ex-1001, 5:8-12. As shown above in Figure 9, proxy 225 appears as a "a single, monolithic application program and all details of the components which make up the application program are essentially hidden" from the renderer because the renderer only receives the "reformatted Web document" from proxy 225. Beranek's proxy 225 is "a piece of standalone or downloadable code that intercepts and filters the Web document retrieved over the computer network." Ex-1005, 12:14-16. Proxy 225's functions of filtering, managing/processing, and monitoring the network-retrieved data and controlling the "look and feel" of the components that make up the application program on the browser interface are hidden from the renderer by being located in or behind proxy 225. Ex-1005, 8:63-67.

For example, as shown below in Beranek's Figure 7, the proxy intercepts a request from the browser (i.e., the claimed renderer) at step 254 and performs the proxy's functions (steps 256, 258, 266, 268, 269, and 271, purple) before returning the web data to the browser; "[t]he ***browser is not 'aware'*** that the GET request has been serviced locally." *Id*., 11:33-50. Therefore, the proxy implements "'dynamic HTML' processing or 'filtering' of the Web document," but the proxy does not provide those details to the renderer to render the Web document on the browser at step 264. Ex-1005, 9:5-7 ("This operation may be effected without the browser being aware that the service is being handled locally."); Ex-1002 ¶109.



Beranek teaches that the proxy is defined to determine changes required to be made to the GUI for the application program during execution of the application program, where the given change is rendering a new layout in the GUI and updating a present layout in the GUI. Ex-1002 ¶110. Beranek's proxy includes a filter mechanism to "re-format a retrieved HTML document before the document is presented on the browser." Ex-1005, 9:8-10. The proxy is used to modify and/or process data before it is displayed on the user's interface, such that the "data font or presentation to be modified, supplemented or otherwise adapted to the particular display environment." *Id*., 12:17-19. Specifically, as shown below in Figure 6, the proxy determines the HTML tags from the Web document being parsed and to remove, replace, or modify tag attributes (162, 166). *Id*., 10:42-11:3. Beranek's proxy determines a change in rendering and updating the UI with a new layout when determining to remove the tag and save the HTML information for inclusion as normal text to support the frame in the client's display (*id*., 10:58-65) and/or determines whether to "change document foreground or background color…, the color of a particular image…, or the font type or size of certain text." *Id*., 11:4-7. Beranek's proxy thus provides a tool to enhance and tailor the "look and feel" of the layout of the data displayed and update the "user interface to be tailored to the Web appliance environment." Ex-1005, 12:19-23.



d. Elements 1[D]/9[C]: [operating] a workflow manager defined to communicate one or more document viewers and associated user interface components to the proxy for provision to the renderer as components necessary for rendering the GUI[,]/[.]

Beranek-Pleyer teaches elements 1[D]/9[C]. Ex-1002 ¶¶111-119.

Beranek-Pleyer teaches operating a workflow manager defined to communicate one or more document viewers and associated user interface components to the proxy for provision to the renderer as components for rendering the GUI. As discussed in Section X, for the purposes of this proceeding, Petitioner has applied this limitation to include software that operates in the same way as "workflow manager 216," which "manages communication between document viewers 212, user interface components 208 and, through application proxy 204, with renderer 200." Ex-1001, 5:57-59, Figure 2; Ex-1002 ¶112.

As shown in Figure 9, Beranek teaches that its proxy is given the Web document to filter and format by a data stream 180, received from the computer network. Ex-1005, 13:6-16; Ex-1002 ¶¶113-114.

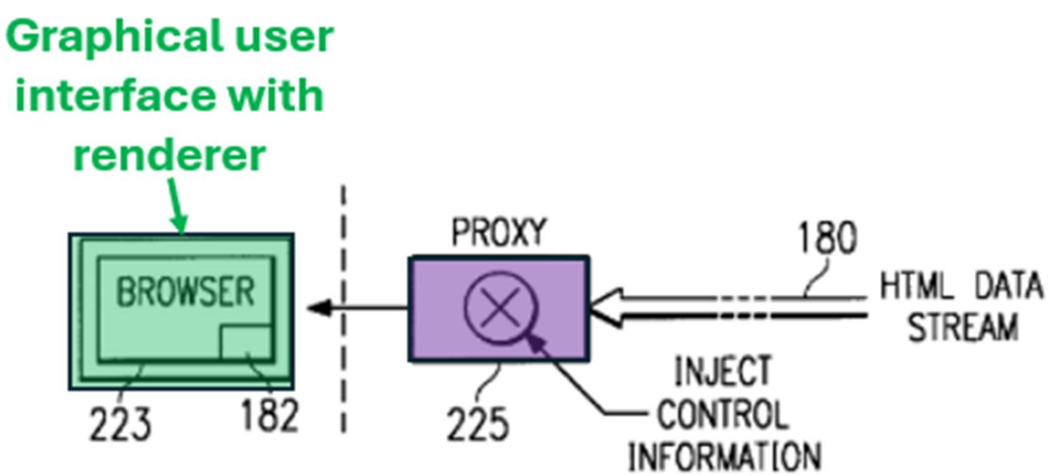


FIG. 9

Similarly in Figure 10 (below), Beranek teaches that in step 184, the data stream is received from "a server at which the document is supported" and in step 186, the stream is intercepted and parsed prior to its delivery to the browser. Ex-1005, 13:24-26.

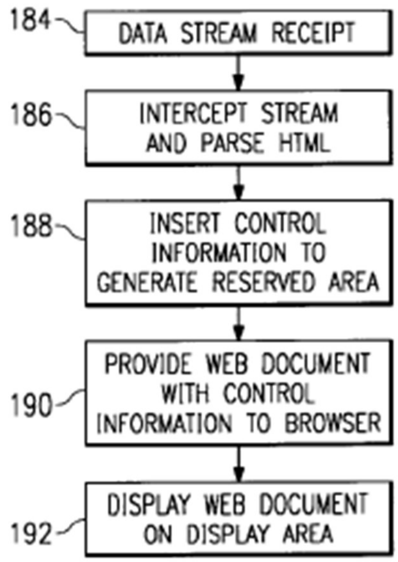


FIG. 10

A POSITA would have been motivated to implement a workflow manager with Beranek's method, as taught by Pleyer, to manage communications of the document viewers and user interface components to be rendered to the proxy. Ex-1002 ¶¶115-119. As shown below in Figure 3, Pleyer teaches a supervisory application 100 that "communicates with hardware devices such as the network interface television tuner, keyboard, and display components though conventional Windows operating system calls." Ex-1006, 5:55-65. The supervisory application communicates the global script 106, which includes browsable content 104 and UI components 105.

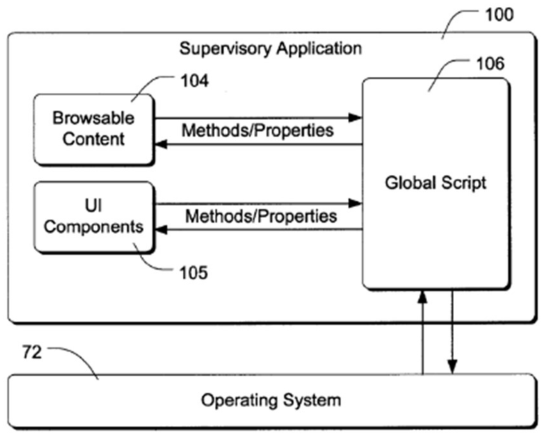


Fig. 3

The Beranek-Pleyer combined system would thus take the form depicted below, in Figure 9':

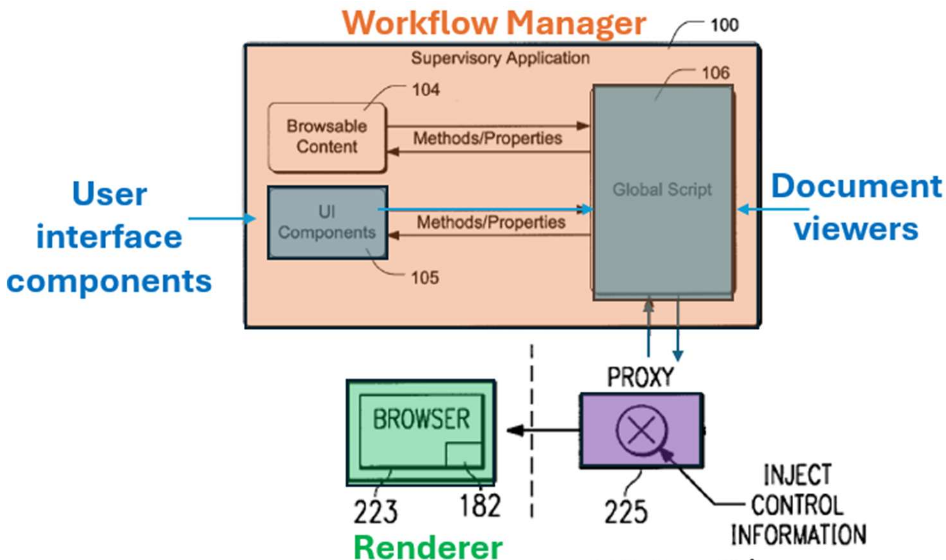


FIG. 9'

Pleyer's workflow manager (supervisory application 100) communicates one or more document viewers implemented as a global script. Ex-1002 ¶117. Pleyer teaches that "visual components of a user interface [are] implemented through a combination of documents or source files written in conventional internet hypertext formats." Ex-1006, 1:60-64. For example, hypermedia content, such as "text, images, sounds, and actions" are provided to a user as "individual 'documents' or 'pages' with embedded control information." *Id*., 3:1-5, 3:10-13. Pleyer teaches that "controls and dialogs … are implemented as text and graphics documents written in … HTML," and that in order to present options to a user, "***global script 106*** generates dialog boxes and controls by specifying local HTML graphical elements." *Id*., 7:40-45. The global script generates user interface components using the "ShowModalDialog(), ShowModelessDialog( ), CloseModalDialog( ), and CloseModelessDialog( ) methods, referred to herein as 'open' and 'close' interface methods." *Id*., 8:56-60. The global script is the document viewer that the workflow manager communicates to the proxy to render the document for the user's observation and interaction with data appropriate to the application program, in all its possible contexts. Ex-1002 ¶¶117-118.

Pleyer teaches that the document viewer includes associated user interface elements. Specifically, Pleyer teaches that "appropriate HTML elements" are displayed to the user for the user to interact with device hardware. Ex-1006, 2:4-10. Pleyer teaches that "executable programs, text or multimedia documents, sound clips, audio segments, still images, computers, directors, and other hyperlinks" may be rendered in the displayed document. *Id*., 3:47-55. The formatting of the document includes "HTML markup codes or 'tags,'" which instruct the Web browser "how to render and print documents." *Id*., 4:1-5. JavaScript is used to "add interactivity to HTML, to allow for user interaction and feedback, multimedia and animation, and to link HTML to other technologies such as Java, Active X, and plug-ins." *Id*., 4:12-16. For example, Pleyer teaches that "event handlers implemented by global script 106" includes generating an "on-screen keyboard" and "audio rendering" (*id*., 9:1-3, 9:29-30), and "create a new user interface component in response to a user pressing a specific key." *Id*., 9:49-51.

A POSITA would have been motivated to implement Beranek-Pleyer together so that the workflow manager communicates one or more document viewers with associated user interface components, implemented by a script, as taught by Pleyer, to control the interaction of displayed controls with underlying hardware and display appropriate graphic elements and for the additional reasons provided in Section XII.A.1. Ex-1006, 10:35-40; Ex-1002 ¶119.

e. Element 1[E]: wherein the renderer, the proxy, and the workflow manager are executed from the non-transitory computer readable medium.

Beranek-Pleyer teaches element 1[E]. Ex-1002 ¶¶120-121.

Beranek teaches that its software components, e.g. the renderer, proxy, and workflow manager, are executed from the non-transitory computer readable medium. Beranek teaches "[a] computer program product in computer-readable media for use in controlling presentation on a client of a Web document formatted according to a markup language (HTML) and supported on a server, the client including a browser and connectable to the server via a computer network." Ex-1005, 16:43-47 (Cl. 19). Beranek also teaches that the "preferred implementations of the client side or server side proxy … is as a set of instructions (program code)." *Id*., 14:63-66. The instructions may be "stored in another computer memory, for example, in a hard disk drive, or in a removable memory such as an optical disk (for eventual use in a CD ROM) or floppy disk (for eventual use in a floppy disk drive), or downloaded via the Internet or other computer network." *Id*., 14:66-15:5. Beranek also teaches that "one of ordinary skill in the art would also recognize such method may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps." *Id*., 15:8-10.

3. Dependent Claims 2-5 and 10-13

a. Claim 2: A computer implemented system for providing a graphical user interface as recited in claim 1, wherein the proxy enables details of each component which makes up the application program to be hidden from the renderer.

Claim 10: A method for providing a graphical user interface as recited in claim 9, wherein the proxy functions to hide details of each component which makes up the application program from the renderer.

Beranek-Pleyer teaches Claims 2 and 10. Ex-1002 ¶¶122-125.

Beranek-Pleyer teaches that the proxy hides details of each component which makes up the application program from the renderer. As discussed in Section X, for the purposes of this proceeding, Petitioner has applied this limitation to include that the proxy's component functionality "is not visible to the user." Ex-1001, 5:23-25; Ex-1002 ¶123.

Beranek teaches that its proxy hides details of each component which makes up the application program from the renderer as explained in Section XII.A.2.c (elements 1[C]/9[B]). Specifically, Beranek's proxy 225 appears just as "application proxy 204," which is "a single, monolithic application program and all details of the components which make up the application program are essentially hidden from renderer 200 which need only interface to application proxy 204." Ex-1001, 5:8-12. For example, Beranek's proxy 225's components are hidden from the renderer because the renderer "***is not 'aware'***" how its request was serviced by the proxy. Ex-1005, 11:33-50. The renderer receives the "reformatted Web document" from the proxy after it is reformatted by the proxy's components without the details of how the Web document was reformatted. Therefore, while the proxy implements "'dynamic HTML' processing or 'filtering' of the Web document," the proxy does not provide those details to the renderer to render the Web document on the browser. Ex-1005, 9:5-7 ("This operation may be effected ***without the browser being aware*** that the service is being handled locally."); Ex-1002 ¶¶124-125.

b. Claim 3: A computer implemented system for providing a graphical user interface as recited in claim 1, wherein the proxy enables each component which makes up the application program to be modified in a manner that is transparent to the renderer.

Claim 11: A method for providing a graphical user interface as recited in claim 9, wherein the proxy functions to enable each component which makes up the application program to be modified in a manner that is transparent to the renderer.

Beranek-Pleyer teaches Claims 3 and 11. Ex-1002 ¶¶126-130. Beranek-Pleyer-Carter teaches that the proxy enables each component which makes up the application program to be modified in a manner transparent to the renderer. As discussed in Section X, PO stated during a meet-and-confer that it interpreted "transparent to the renderer" as requiring that the renderer is not aware of any modification of "each component which makes up the application program." For completeness, Petitioner address PO's proposed alternative construction here. Petitioner's proposed alternative construction is presented in Ground 3.

Under PO's interpretation, Beranek-Pleyer teaches Claims 3 and 11. Beranek teaches that its proxy enables components of the application program to be modified in a manner that is transparent to the renderer while the renderer is not aware of any modification. As described in Section XII.A.2.c (elements 1[C]/9[B]), Beranek's proxy 225 hides the components of the application program from the renderer because the renderer "***is not 'aware'***" how its request was serviced by the proxy. Ex-

1005, 11:33-50; *see also id*., 9:1-7. The renderer receives the "reformatted Web document" (i.e., components of the which makes up the application program) from the proxy, after it is reformatted by the proxy without the details of how the Web document was reformatted, so that Beranek's proxy 225 appears as "a single, monolithic application program and all details of the components which make up the application program are essentially hidden." *See supra* section XII.A.2.c (elements 1[C]/9[B]). Because the renderer "is not 'aware'" how its request is serviced, the renderer is able to modify the application program's components without being aware of the proxy's reformatting the Web document/application program. Ex-1002 ¶¶128-129.

For example, Beranek's proxy enables components of the application program to be modified while being invisible to the renderer. Ex-1002 ¶¶129-130. The '428 Patent explains that the proxy enables the "components of the application program to be modified in a manner which is transparent to the renderer" and provides an example of "an improved 'Edit Picture' document viewer… substitut[ed] in the application program." Ex-1001, 5:12-17. Just as the improved "Edit Picture" document viewer is rendered without the renderer being aware of how the proxy handled the modification, Beranek teaches that its proxy updates the Web document without making the details of the updating known to the renderer. Ex-1002 ¶¶129-130. Specifically, Beranek's DHTML browser allows for the renderer to display dynamic HTML functions, allowing a web page's content, style, and structure to be edited after the web page has been loaded, without requiring a full page reload. *Id*. As shown below in Figure 5, when the proxy determines that the DHTML function is active in step 235, the "proxy identif[ies] the HTML format and other information therein that may need to be ***supplemented, modified or otherwise changed*** to affect the appearance of the document" in step 237. Ex-1005, 10:13-20. The proxy filters the parsed information "according to a given filter property," "***defined by the system" or "defined by the user using the remote control***." *Id*., 10:20-24. The document is modified and passed to the renderer according to the filtering defined by the system or the user. *Id*., 10:25-27. Therefore, the proxy enables the components of the application program (the rendered Web document) to be edited in a manner that is transparent to the renderer by following filtering rules defined by the renderer, while hiding the details of how the edit is performed from the renderer. Ex-1002 ¶130. Beranek's proxy remains invisible to the renderer but allows for components of the application program to be modified in a manner transparent to the renderer. *Id*.

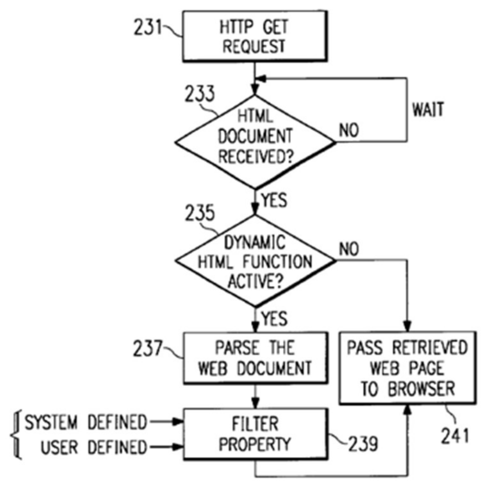


FIG. 5

c. **Claim 4: A computer implemented system for providing a graphical user interface as recited in claim 1, wherein each of the one or more document viewers includes functions to enable the renderer to display a current document in an appropriate format.**

Claim 12: A method for providing a graphical user interface as recited in claim 9, wherein each of the one or more document viewers includes functions to enable the renderer to display a current document in an appropriate format.

Beranek-Pleyer teaches Claims 4 and 12. Ex-1002 ¶¶131-133.

Beranek-Pleyer teaches that its document viewers include functions to enable the renderer to display a current document in an appropriate format. Beranek teaches one or more document viewers, which include functions to enable the renderer to display a current document in an appropriate format by identifying the format of the Web document as it is received from the server (Ex-1005, 3:2-5) and "***re-format*** the Web document according to some given protocol or filter property." *Id.*, 3:5-8. For example, Beranek teaches "adjust[ing] or modify[ing] foreground/background colors, overrid[ing] font types and size, control[ing] display geometry (e.g., by reserving some dedicated display area), chang[ing] URLs or display[ing] other URLs, embed files, add, remove or reconfigure frames, or control text and table formats." *Id*., 3:5-14. Beranek thus includes functions to enable the renderer to display the document in an appropriate format after the proxy has reformatted the received Web document from the data stream. Ex-1002 ¶132.

A POSITA would have been motivated to implement Beranek's document viewers to include functions to enable the renderer to display a current document in an appropriate format, as detailed by Pleyer. Pleyer teaches that "visual components of a user interface [are] implemented through a combination of documents or source files written in conventional internet hypertext formats." Ex-1006, 1:60-64. For example, hypermedia content, such as "text, images, sounds, and actions" are provided to a user as "individual 'documents' or 'pages' with embedded control information." *Id*., 3:1-5, 3:10-13. Pleyer teaches that "controls and dialogs … are implemented as text and graphics documents written in … HTML," and that in order to present options to a user, "***global script 106*** generates dialog boxes and controls by specifying local HTML graphical elements." *Id*., 7:40-45. The global script generates user interface components using the "ShowModalDialog( ), ShowModelessDialog( ), CloseModalDialog( ), and CloseModelessDialog( )

methods, referred to herein as 'open' and 'close' interface methods." *Id*., 8:56-60. Pleyer's global script includes functions for the renderer to display a current document in an appropriate format because it communicates what to render for the user's observation and interaction with data appropriate to the application program, in all its possible contexts. Ex-1002 ¶133.

d. Claim 5: A computer implemented system for providing a graphical user interface as recited in claim 1, wherein each document viewer includes a respective user interface requirements specification referencing one or more user interface components associated with the document viewer.

Claim 13: A method for providing a graphical user interface as recited in claim 9, wherein each document viewer includes a respective user interface requirements specification referencing one or more user interface components associated with the document viewer.

Beranek-Pleyer teaches Claims 5 and 13. Ex-1002 ¶¶134-136.

Beranek-Pleyer teaches that each document viewer includes a respective user interface requirements specification referencing one or more user interface components associated with the document viewer. Specifically, Pleyer teaches user interface components such as "options" are presented to a user by the global script 106 generating "dialog boxes and controls by specifying local HTML graphical elements … to the supervisory application" which then "passes" such "***specifications*** to a browser…, [to] display[] the specified HTML graphical elements in conjunction with whatever material is already being displayed as a result of user browsing." Ex-1006, 7:41-49. Pleyer teaches specific events and the resulting behavior to be handled by the global script. *Id*., 9:1-31. For example, in response to a user pressing a specific key, Pleyer implements an "event handler in the global script" to create a new user interface component, by showing a toolbar and dialog window. *Id*., 9:47-63. In another example, a "TVTuner component" can be implemented as a user interface component rendered by displaying an "ActiveX TV Tuner control as specified by the class identifier (CLASSID)" so that a "TV viewing window in response to a user pressing a specific key" is opened. *Id*., 9:64-10:15. Pleyer's "event handlers" are the respective user interface requirements specification that reference one or more user interface components associated with the document viewer, the global script, because the event handlers are respective user interface requirements specifications handled by the global script according to the requirements for that event. Ex-1002 ¶136.

B. Ground 2: Claims 3 and 11 are obvious in view of Beranek-Pleyer in view of Carter.

1. A POSITA would have been motivated to combine Beranek-Pleyer's teachings and Carter's and would have had a reasonable expectation of success.

A POSITA would have been motivated to combine Beranek-Pleyer's teachings and Carter's teaching and had a reasonable expectation of success in doing so because each relates to controlling how a document is presented for display on a browser of a client machine. Ex-1005, Title, Abstract; Ex-1006, Title, Abstract; Ex-1007, Abstract; Ex-1002 ¶¶137-145.

As discussed above, Beranek teaches "[a] method of controlling presentation on a client of a Web document formatted according to a markup language and supported on a server, the client including a browser and connectable to the server via a computer network." Ex-1005, 15:33-36 (Cl. 1). Beranek intercepts, reformats, and controls how a Web document appears in the browser. Ex-1005, 2:54-61. Specifically, a "client-side caching proxy," "intercept[s] the Web document and then dynamically rewrite[s] the document before it is displayed on the browser." Ex-1005, 2:63-3:2. In Beranek-Pleyer's system "[t]he ***browser is not 'aware'***" of how the user's request is serviced by the proxy because the functions are hidden. Ex-1005, 11:33-50.

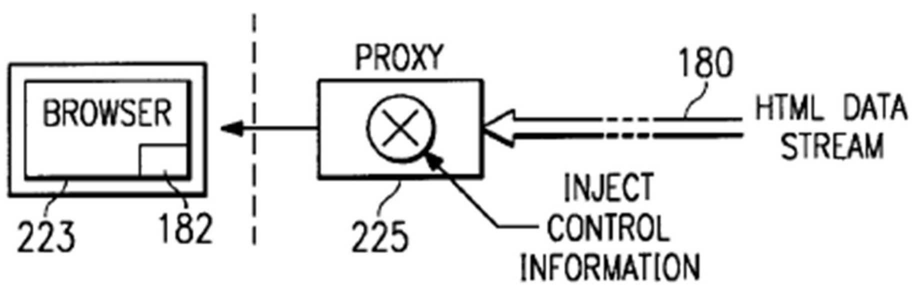


FIG. 9

Carter teaches "dynamically construct[ing] a graphical user interface associated with an application" by determining which command classes to "instantiate" and constructing "the graphical user interface for the application such that the graphical user interface contains the additional functions of the successfully instantiated command classes." Ex-1007, Abstract. Carter teaches a constructor 3120, which "locates any packages (or classes in a package)." Ex-1007, 5:27-37. The program will determine "which classes are available and then construct a graphical user interface ***to reflect all of the classes***." *Id*., 2:56-57. Figure 3 shows a command class library which stores multiple command classes 3100-3400 that include additional application functionality. *Id*., 4:27-30. Carter's command classes are rendered on the application so that the user may select the additional functionality, which then causes a new layout in the GUI and updating a present layout in the GUI for the user. *See, e.g.*, Figs. 5A, 5B.

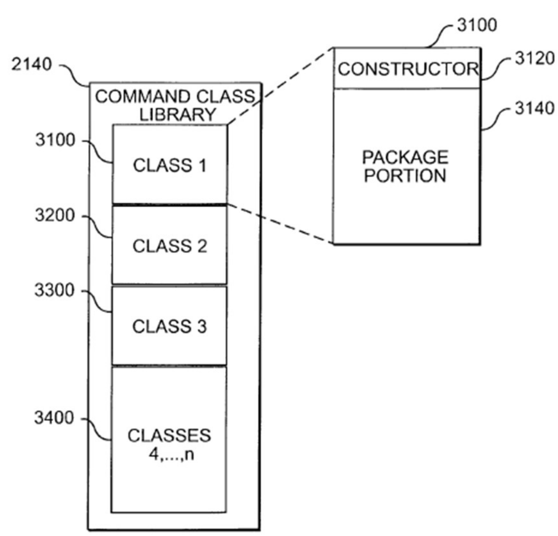
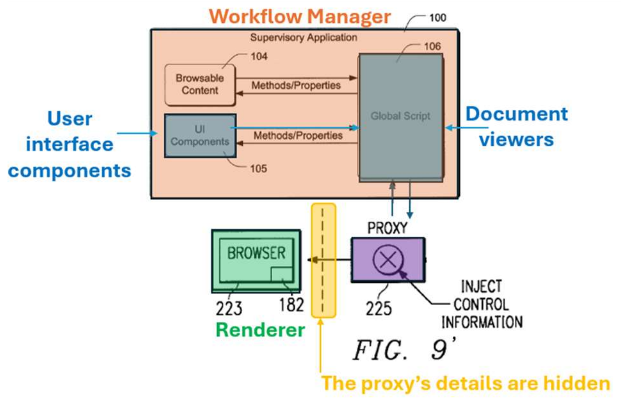
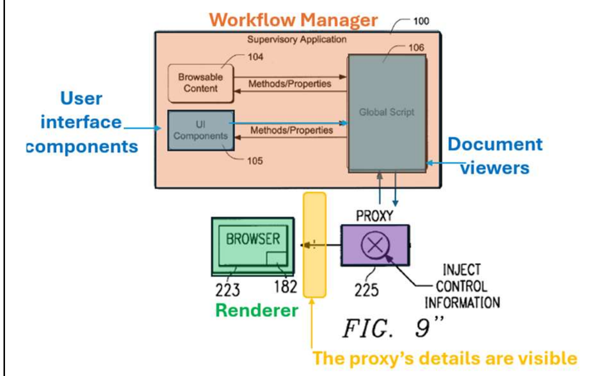


FIG. 3

A POSITA would have been motivated for Beranek-Pleyer's proxy functionality to be transparent to the renderer, and thus the user, as taught by Carter. so that they can be invoked, managed, and undone through a consistent interface directly by the user. Ex-1002 ¶¶138-143. Carter determines the form of a graphical user interface associated with an application based on a run-time determination of the available features in order to "provide only those additional functions that the customer desires" without "the creation of several versions of the application." Ex-1007, 1:56-58, 2:1-3, 6:15-18.

A POSITA would have had a reasonable expectation of success in combining the teachings of Beranek-Pleyer and Carter. Ex-1002 ¶¶142-145. Indeed, as shown below in Modified Figure 9'' below, Carter's command classes (that add functionality) enable the functions that are performed by Beranek's proxy to be modified in a manner that is transparent to the renderer because the user can see the available options on how to change the layout of the GUI so that the line between the renderer and proxy is removed. Ex-1002 ¶¶144-145.

Beranek-Pleyer	Beranek-Pleyer-Carter
	

Moreover, the references describe methods for presenting a requested web document to a user in an appropriate format on a graphical user interface by the user using similar hardware and software components. Ex-1005, Title, 15:33-36 (Cl. 1); Ex-1006, Title, Abstract; Ex-1007, Abstract. A POSITA would thus have understood that these references disclose interrelated teachings based on well-understood technologies that would have been amenable to various well-understood and predictable combinations. Ex-1002 ¶145.

2. Dependent Claims 3 and 11

a. Claim 3: A computer implemented system for providing a graphical user interface as recited in claim 1, wherein the proxy enables each component which makes up the application program to be modified in a manner that is transparent to the renderer.

Claim 11: A method for providing a graphical user interface as recited in claim 9, wherein the proxy functions to enable each component which makes up the application program to be modified in a manner that is transparent to the renderer.

Beranek-Pleyer-Carter teaches Claims 3 and 11. Ex-1002 ¶¶146-150. Beranek-Pleyer-Carter teaches that the proxy enables each component which makes up the application program to be modified in a manner transparent to the renderer. As discussed in Section X, under an alternative construction, Petitioner has applied this limitation to mean that the proxy's component functionality is "transparent to the renderer" such that the renderer can see through the proxy to additional control options that may be available. Ex-1001, 5:12-14.

While Beranek explains that its renderer is unaware of how the Web document is provided from the proxy, a POSITA would have been motivated for Beranek-Pleyer's proxy to function so that each component which makes up the application program is transparent to the renderer, such that the renderer can see through the proxy to additional control options that may be available, as further detailed in Carter. Ex-1002 ¶148. Carter teaches a constructor 3120 which "locates any packages (or classes in a package)." Ex-1007, 5:27-37. Carter's constructor program determines "which classes are available and then construct[s] a graphical user interface ***to reflect all of the classes***." *Id*., 2:56-57. Figure 3 shows a command class library which stores multiple command classes 3100-3400 that are additional application functionality. *Id*., 4:27-30.

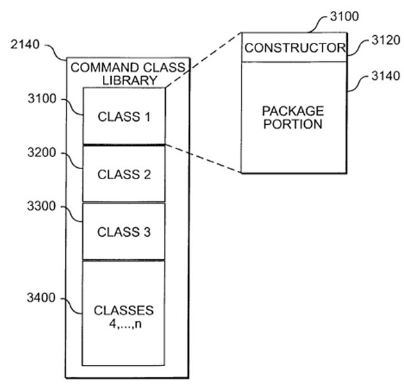


FIG. 3

In Figure 5B, a GUI presents four basic controls of "In Box 5210, Out Box 5220, Trash 5230, and Send Mail 5240" and additional controls for a mail application, by instantiating one or more command classes, such as "a Spell Checker 5250 and a Thesaurus 5260." *Id*., 5:47-56. Carter's proxy application program components are thus visible to the renderer because the proxy is transparent such that available options/classes can be seen by the renderer and can be displayed to be selected by the user. Ex-1002 ¶¶148-149. While the actual underlying details of how the software is implemented and changes made are hidden from the renderer, the control options that drive the software to make those changes are exposed to the user. As such, the proxy hides specific software details from the user but is transparent to how those details may be defined and controlled. Ex-1002 ¶149.

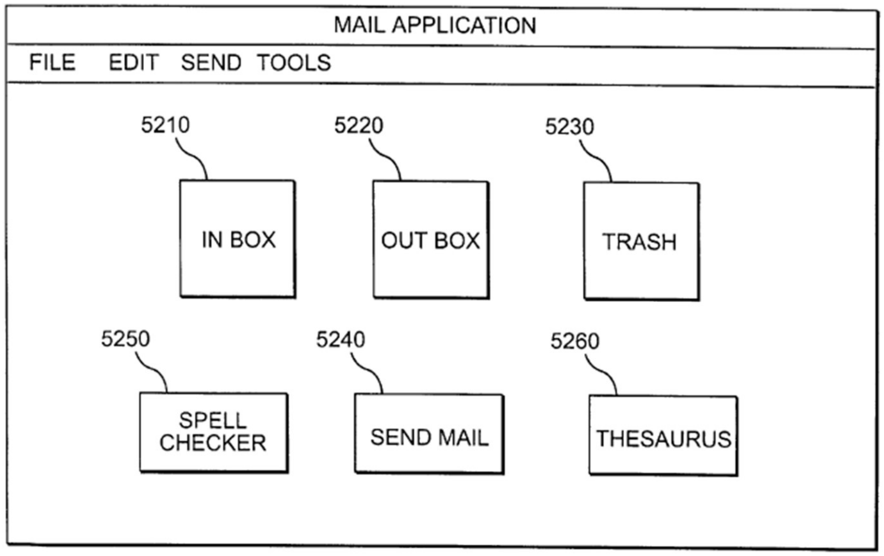


FIG. 5B

A POSITA would have been motivated for Beranek-Pleyer's proxy components functions to be modified in a manner that is transparent to the renderer, as taught in Carter, so that additional features can be invoked, managed, and undone through a consistent interface directly by the user which increases functionality that may be available to the user, but does not burden the renderer with the actual implementation details of those controls. Ex-1002 ¶150.

C. Ground 3: Claims 9-11 are obvious over Ditmer-Beranek

1. A POSITA would have been motivated to combine Ditmer's teachings with Beranek's teachings and would have had a reasonable expectation of success.

A POSITA would have been motivated to combine the teachings of Ditmer and Beranek and had a reasonable expectation of success in doing so because each relates to using a proxy to provide data in response to user requests via a browser/GUI. Ex-1008, Title, Abstract, Figure 6; Ex-1005, Title, Abstract; Ex-1006, Title, Abstract; Ex-1002 ¶¶151-157.

Ditmer teaches a "Web-based fault and alarm management tool that enables customers to monitor and analyze the performance of their voice and data network via a graphical user interface." Ex-1008, Abstract; *see also id.*, 3:27-36. Ditmer provides a "system and a method for communicating information relating to the dedicated voice and data networks, from an enterprise to a customer at a client workstation, the present invention includes a client browser application located at the client workstation." *Id.*, 3:52-56. As shown below in Figure 6, Ditmer teaches that a user's request is forwarded to "an integrated proxy application 640 [(purple)] for receiving and interpreting the user messages and enabling the event monitor functionality." Ex-1008, 14:31-34. Specifically, the proxy confirms incoming requests and "manages the communication of the specific customer request with the middle-tier server to actually get the request serviced." *Id.*, 23:7-14; *see also id.*, 24:14-20. The "[d]ata returned from the … server is translated back to client format, if necessary, and returned to the dispatch server as a response to the request." *Id.*, 23:14-17; *see also id.*, 9:46-50. Ditmer's "event monitor server 650 [(orange)] is responsible for all tasks leading up to and including the management of alarms and performance reports including data collection, calculation, storage, and report generation." Ex-1008, 14:58-65.

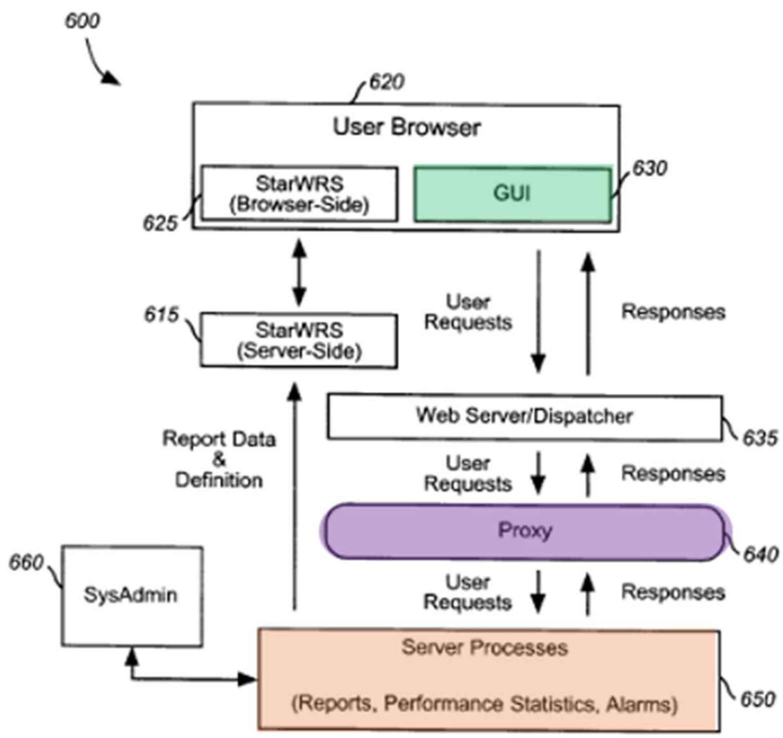


FIG. 6

Beranek teaches "[a] method of controlling presentation on a client of a Web document formatted according to a markup language and supported on a server, the client including a browser and connectable to the server via a computer network." Ex-1005, 15:33-36 (Cl. 1). Beranek's proxy intercepts, reformats, and controls how a Web document appears in the browser. Ex-1005, 2:54-61, 2:63-3:2. The proxy will "re-format the Web document according to some given protocol or filter property," such as "adjust[ing] or modify[ing] foreground/background colors, overrid[ing] font types and size, control[ing] display geometry (e.g., by reserving some dedicated display area), chang[ing] URLs or display[ing] other URLs, embed files, add, remove or reconfigure frames, or control text and table formats." *Id.*, 3:5-14. The "reformatted Web document is then passed to the browser for display." Ex-1005, 3:7-8, Abstract. Beranek teaches that, as the Web document is received from the server, "the HTML is parsed to identify the format of the document and the information therein." Ex-1005, 3:2-5.

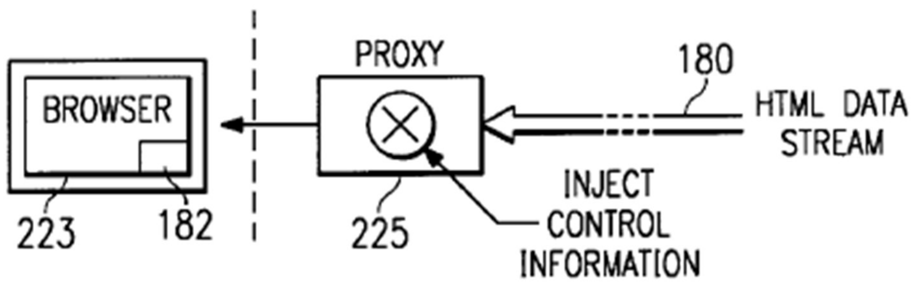


FIG. 9

A POSITA would have been motivated to implement Beranek's proxy technique of formatting the document to be rendered on the user's GUI into Ditmer's proxy to ensure that the web document/data that is sent back to be rendered on the user's GUI is translated back to a format that can be displayed. While Ditmer teaches that its proxy may generate web pages based on alarm state and other data reports received from the server, Beranek explicitly teaches determining the GUI changes to be made, such as determining the GUI's new layout and updating the present layout in the GUI. Ex-1002 ¶¶152-154. A POSITA would have been motivated to implement Beranek's proxy, which determines the changes to be made, into Ditmer for Beranek's added benefits of its proxy being used to "modify and/or process data before it is returned … for display on the interface," which provides a "useful tool to enhance the 'look and feel' of data displayed… and thus enables the user interface to be tailored to the Web application environment." Ex-1005, 12:16-18; *see also id*., 14:10-13 ("optimize 'look and feel'"). Moreover, implementing Beranek's teachings provides "dynamic[] control [of] how information is presented on a… client machine" such that it enables "content modification at a client machine" while "an author of such content need only create a single version of the page." *Id*., 2:12-14, 2:19-21. Beranek's teaching also provides "an enhanced display presentation capability" through the use of its proxy. *Id*., 2:44-47. A POSITA would thus have been motivated to implement Ditmer with Beranek's teachings to achieve these additional benefits. Ex-1002 ¶154.

A POSITA would also have been motivated to ensure that the data provided by Ditmer's process server includes one or more document viewers and associated user interface components, as taught by Beranek, so that the user may interact with the web document by providing the framework to display user interface elements, such as buttons, sliders, and sound on a web document. Ex-1002 ¶¶155-156.

A POSITA would have had a reasonable expectation of success in combining the teachings of Beranek and Pleyer. Ex-1002 ¶157. Moreover, both references describe methods for presenting requested data to a user in an appropriate format on a graphical user interface by the user using similar hardware and software components. Ex-1005, Title, 15:33-36 (Cl. 1); Ex-1008, Abstract. A POSITA would thus have understood that these references disclose interrelated teachings based on well-understood technologies that would have been amenable to various well-understood and predictable combinations. Ex-1002 ¶157.

2. Independent Claim 9

a. Element 9[pre]: A method for providing a graphical user interface, comprising:

Ditmer-Beranek teaches the preamble of Claim 9, to the extent that it is limiting. Ex-1002 ¶¶158-159. Ditmer teaches a computer-implemented system for providing a graphical user interface (GUI). Ditmer teaches "an integrated Web-based graphical, user-friendly interface to receive information on the performance of their dedicated voice and data networks." Ex-1008, 3:27-36. Specifically, Ditmer provides a "system and a method for communicating information relating to the dedicated voice and data networks, from an enterprise to a customer at a client workstation, the present invention includes a client browser application located at the client workstation." *Id.*, 3:52-56; Ex-1002 ¶159.

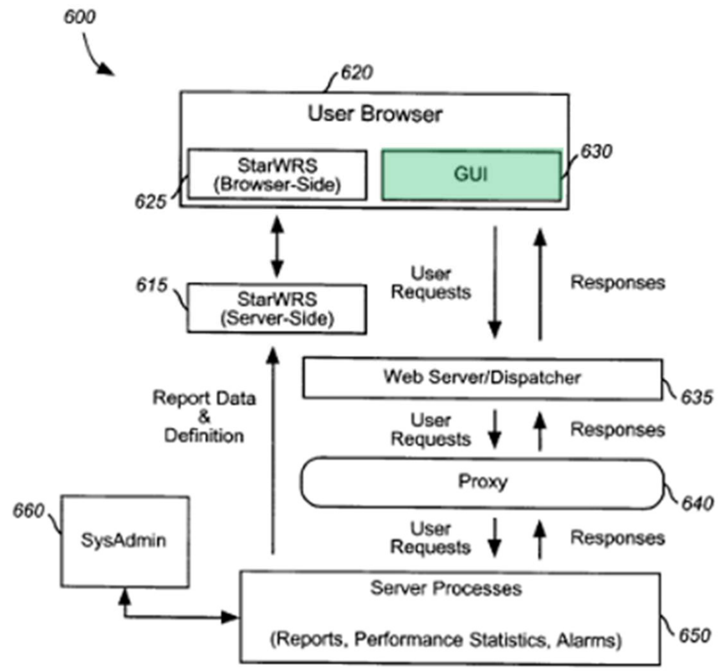


FIG. 6

b. Element 9[A]: operating a renderer defined to render a graphical user interface (GUI) for an application program;

Ditmer-Beranek teaches element 9[A]. Ex-1002 ¶¶160-163.

Ditmer-Beranek teaches operating a renderer defined to render a GUI for an application program. As discussed in Section X, for the purposes of this proceeding, Petitioner has treated this limitation as including software that renders a graphical user interface for an application program that same way as the patent, which would include software such as "a DHTML browser." Ex-1001, 4:53, 5:23-24, 7:19; Ex-1002 ¶161.

Ditmer-Beranek's method operates a renderer to render a GUI for the application program. Ditmer "includes a client browser application located at the client workstation," which "enables interactive Web-based communications … and also provides an integrated interface to the various enterprise application services" by invoking "a Java application/applet." Ex-1008, 3:54-62. Ditmer states that customer browser includes "***client applications responsible for presentation*** and front-end services." *Id*., 6:14-16; *see also id*., Cl. 1 ("presentation device for enabling the customer to define and display reports…"), Cl. 17 (presentation device includes a GUI and may launch directly from the client browser application). A POSITA would have understood that Ditmer's client application, responsible for presentation and front-end services on a browser, is a renderer that renders HTML pages for alarm management tools because it displays the application program on the user's workstation GUI. Ex-1002 ¶162.

While Ditmer teaches that its client application is a renderer, Beranek explicitly teaches a DHTML browser as a renderer, consistent with how the '428 Patent explains its renderer, as described above in Section XII.A.2.b (element 9[A]). Ex-1002 ¶163. A POSITA would have been motivated to include Beranek's DHTML browser as Ditmer's renderer to dynamically present the user's request. *Id*.

c. Element 9[B]: [(i)] operating a proxy defined to provide each component necessary for rendering the GUI to the renderer, [(ii)] wherein the proxy appears

to the renderer as a single monolithic application program, and [(iii)] wherein the proxy is defined to determine changes required to be made to the GUI for the application program during execution of the application program, [(iv)] wherein a given change determined by the ox is one of rendering a new layout in the GUI and updating a present layout in the GUI; and

Ditmer-Beranek teaches element 9[B]. Ex-1002 ¶¶164-168.

Ditmer-Beranek teaches operating a proxy defined to provide each component necessary for rendering the GUI to the renderer. Ex-1002 ¶165. Ditmer teaches that a user's request is forwarded to "an integrated proxy application 640 for receiving and interpreting the user messages and enabling the event monitor functionality." Ex-1008, 14:31-34. The proxy 640 "sends the generic header followed by the proxy-specific response back… to the user browser 620." *Id.*, 14:42-46. Specifically, the proxy examines and confirms incoming requests to ensure that "they include validly formatted messages for the service with acceptable parameters; a translation process for translating a message into the database query message or networking protocol; and, a management process for managing the communication of the specific customer request with the middle-tier server to actually get the request serviced." *Id.*, 23:7-14; *see also id.*, 24:14-20. The "[d]ata returned from the … server is translated back to client format, if necessary, and returned to the dispatch server as a response to the request." *Id.*, 23:14-17; *see also id.*, 9:46-50.

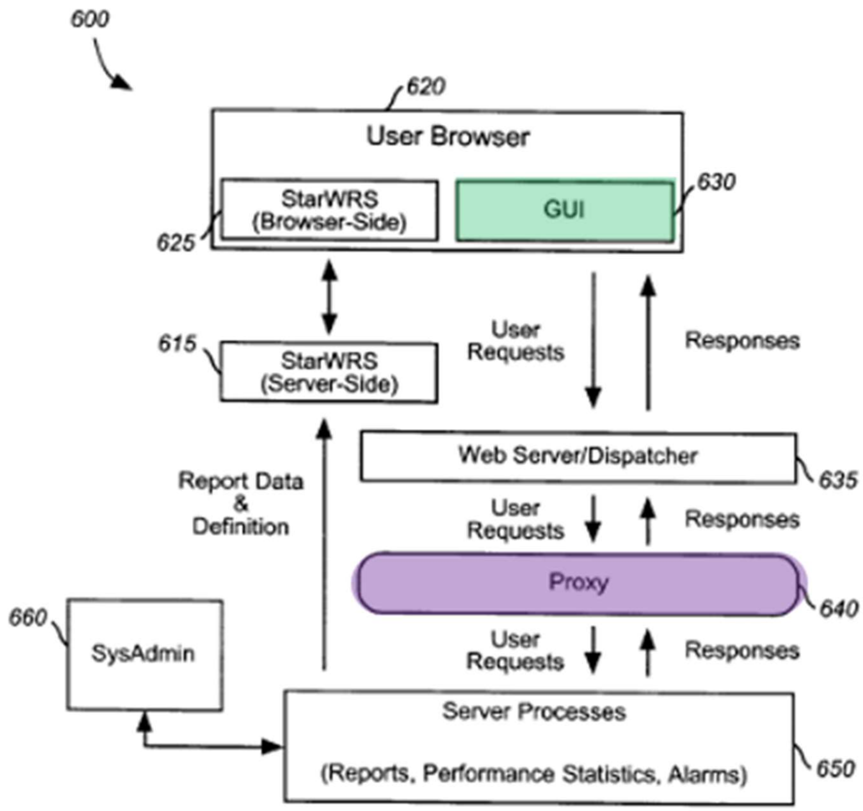


FIG. 6

While a POSITA would have understood that Ditmer's proxy provides each component necessary for rendering the GUI on the user's browser in response to the user's request, Beranek provides further implementation details, as described above in Section XII.A.2.c (element 9[B]). Ex-1002 ¶166.

In Ditmer-Beranek, the proxy appears to the renderer as a single monolithic application program. As discussed in Section X, for the purposes of this proceeding, Petitioner applied this limitation to include software that operates in the same way as "application proxy 204," which is "a single, monolithic application program and all details of the components which make up the application program are essentially hidden from renderer 200 which need only interface to application proxy 204." Ex-1001, 5:8-12. As described in Section XII.A.2.c, Beranek's proxy 225 appears as a "a single, monolithic application program and all details of the components which make up the application program are essentially hidden" from the renderer because the renderer only receives the "reformatted Web document" from proxy 225. Beranek's proxy 225 is "a piece of standalone or downloadable code that intercepts and filters the Web document retrieved over the computer network." Ex-1005, 12:14-16. Proxy 225's functions of filtering, managing/processing, and monitoring the network-retrieved data and controlling the "look and feel" of the browser interface are hidden from the renderer by being located in or behind proxy 225. Ex-1005, 8:63-67; Ex-1002 ¶167.

Ditmer-Beranek teaches that the proxy is defined to determine changes required to be made to the GUI for the application program during execution of the application program, where the given change is rendering a new layout in the GUI and updating a present layout in the GUI. Ex-1002 ¶168. While Ditmer teaches that its proxy may generate web pages based on alarm state and other data, Beranek explicitly teaches determining the GUI changes to be made, such as determining the GUI's new layout and updating the present layout in the GUI, as described above in Section XII.A.2.c (element 9[B]).

d. Element 9[C]: operating a workflow manager defined

to communicate one or more document viewers and associated user interface components to the proxy for provision to the renderer as components necessary for rendering the GUI.

Ditmer-Beranek teaches element 9[C]. Ex-1002 ¶¶169-173.

Ditmer-Beranek teaches operating a workflow manager defined to communicate one or more document viewers and associated user interface components to the proxy for provision to the renderer as components for rendering the GUI. As discussed in Section X, for the purposes of this proceeding, Petitioner has applied this limitation to include software that operates in the same way as "workflow manager 216," which "manages communication between document viewers 212, user interface components 208, and through application proxy 204, with renderer 200."" Ex-1001, 5:57-59, Figure 2; Ex-1002 ¶170.

Ditmer-Beranek teaches operating a workflow manager. Ditmer's "event monitor server 650 is responsible for all tasks leading up to and including the management of alarms and performance reports including data collection, calculation, storage , and report generation." Ex-1008, 14:58-65. As shown in Figure 6 below, event monitor server 650 (orange) is a "workflow manager" because it is a software module that provides report data and definition to the proxy 640 (purple) in response to the user requests for reports. *Id*., 14:54-57; Ex-1002 ¶171.

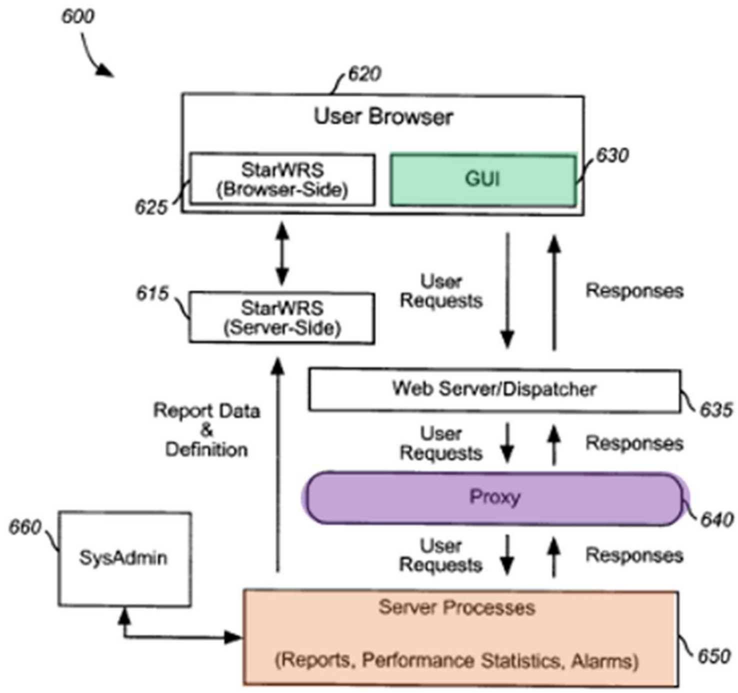


FIG. 6

A POSITA would have been motivated for the Ditmer's event monitor server 650 to communicate one or more document viewers and associated user interface components to the proxy, as taught in Beranek. Ex-1002 ¶172. Beranek's Web document from the server includes one or more document viewers. The '428 specification explains that a document viewer includes "viewers appropriate to observe and interact with data appropriate to the application program, in all of its possible contexts," such as including an "'Edit Picture' viewer, a 'Special Effects' viewer, an 'Object Manager' viewer, etc." Ex-1001, 5:34-41. Beranek teaches that, as the Web document is received from the server, "***the HTML*** is parsed to ***identify the format of the document and the information*** therein." Ex-1005, 3:2-5. The proxy will "***re-format*** the Web document according to some given protocol or filter property," such as "adjust[ing] or modify[ing] foreground/background colors, overrid[ing] font types and size, control[ing] display geometry (e.g., by reserving some dedicated display area), chang[ing] URLs or display[ing] other URLs, embed files, add, remove or reconfigure frames, or control text and table formats." *Id*., 3:5-14. A POSITA would have understood that Beranek teaches a document viewer that communicates the Web document to be rendered via the proxy, which adjusts and modifies the display geometry and adds/removes/reconfigures frames so that the data to be observed and interacted with in all its possible contexts. Ex-1002 ¶172.

Beranek's Web document from the server includes associated user interface components. The '428 specification explains that user interface components "include controls such as button, sliders, radio buttons, edit boxes, etc., can be implemented in a variety of manners, including as ActiveX controls, Java applets, etc." Ex-1001, 4:11-14. As described above, Beranek's proxy will adjust the "look and feel" of a Web document, which is formatted according to a markup language, such as HTML. Specifically, Beranek's proxy will "re-format the Web document according to some given protocol or filter property," such as "adjust[ing] or modify[ing] foreground/background colors, overrid[ing] font types and size, control[ing] display geometry (e.g., by reserving some dedicated display area), chang[ing] URLs or display[ing] other URLs, embed files, add, remove or reconfigure frames, or control text and table formats." Ex-1005, 3:5-14. Beranek's proxy may also "change the font of the data, or to ***add sounds, or to arrange data in a certain tabular format***, or the like." *Id*., 11:42-45. Additionally, Beranek teaches that its proxy's filter mechanism is able to reformat a "***checkbox, button, pulldown menu or the like***," and ***images*** on the web document. *Id*., 9:8-51. A POSITA would have understood that Beranek's Web document, formatted according to HTML, includes associated user interface components within the document viewer so that the user may interact with the Web document because HTML is a common language used to structure content and provides the framework to display user interface elements, such as buttons, sliders, and sound. Ex-1002 ¶173.

3. Dependent Claims 10 and 11

a. Claim 10: A method for providing a graphical user interface as recited in claim 9, wherein the proxy functions to hide details of each component which makes up the application program from the renderer.

Ditmer-Beranek teaches Claim 10. Ex-1002 ¶¶174-176.

Ditmer-Beranek teaches that the proxy hides details of each component which makes up the application program from the renderer. As discussed in Section X, for the purposes of this proceeding, Petitioner has applied this limitation to include that the proxy's component functionality "is not visible to the user." Ex-1001, 5:23-25; Ex-1002 ¶175.

Ditmer-Beranek teaches that its proxy hides details of its components from the renderer as explained in Section XII.B.2.c (element 9[B]). Moreover, for the same reasons described in Section XII.A.3.a, Beranek's proxy 225 appears just as "application proxy 204," which is "a single, monolithic application program and all details of the components which make up the application program are essentially hidden from renderer 200 which need only interface to application proxy 204." Ex-1001, 5:8-12; Ex-1002 ¶176.

b. Claim 11: A method for providing a graphical user interface as recited in claim 9, wherein the proxy functions to enable each component which makes up the application program to be modified in a manner that is transparent to the renderer.

Ditmer-Beranek teaches Claim 11. Ex-1002 ¶¶177-178. Ditmer-Beranek teaches that the proxy enables each component which makes up the application program to be modified in a manner transparent to the renderer. As discussed in Section X, PO stated during a meet-and-confer that it interpreted "transparent to the renderer" as requiring that the renderer is not aware of any modification of "each component which makes up the application program." For the reasons explained in Section XII.A.3.b (Claims 3 and 11), Beranek teaches that its proxy enables each component which makes up the application program to be modified in a manner that is transparent to the renderer in PO's alternative construction.

XIII. CONCLUSION

Petitioner requests institution of IPR for Claims 1-5 and 9-13 of the ’428 Patent based on each of the grounds in this Petition.

Respectfully submitted,

/s/ *Benjamin M. Haber*
Benjamin M. Haber (Reg. No. 67,129)

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner certifies that this petition includes 11,836 words, as measured by Microsoft Word, exclusive of the table of contents, mandatory notices under § 42.8, certificates of service, word count, and exhibits.

CERTIFICATE OF SERVICE (37 C.F.R. § 42.6(e)(1))

The undersigned hereby certifies that the above document was served on November 7, 2025, by filing this document through the Patent Trial and Appeal Board P-TACTS System, as well as delivering a copy via express mail upon the following attorneys of record for the Patent Owner:

Haley Guiliano LLP
75 Broad Street
Suite 510
New York, NY 10004-3222

A courtesy copy was sent to the below counsel via electronic mail:

Bradley W. Caldwell
Jason D. Cassady
John Austin Curry
Brian D. Johnston
Caldwell Cassady & Curry PC
2121 N Pearl St., Suite 1200
Dallas, TX 75201
Telephone: (214) 888-4848
bcaldwell@caldwellcc.com
jcassady@caldwellcc.com
acurry@caldwellcc.com
bjohnston@caldwellcc.com

Brian E. Farnan
Michael J. Farnan
Farnan LLP
919 N. Market St., 12th Floor
Wilmington, Delaware 19801
Telephone: (302) 777-0300
bfarnan@farnanlaw.com
mfarnan@farnanlaw.com

Dated: November 7, 2025

Respectfully submitted,

/s/ Benjamin M. Haber

Benjamin M. Haber (Reg. No. 67,129)
O'Melveny & Myers LLP
400 South Hope Street, 18th Floor
Los Angeles, CA 90071
Telephone: (213) 430-6000
Fax: (213) 430-6407
Email: bhaber@omm.com
Attorney for Petitioner