CLAIM CHART EXHIBIT 6
"WEI94"

INVALIDITY CLAIM CHART FOR U.S. PATENT NO. 5,838,906

- **BASED ON PEI WEI'S PUBLICATION "A BRIEF OVERVIEW OF THE VIOLA ENGINE, AND ITS APPLICATIONS" BEARING THE DATE AUGUST 16, 1994, AVAILABLE AT [PA-00318355]. ALSO AVAILABLE AT [PA-00318385] THROUGH [PA-00318392]. ("WEI94"). THE BODY OF MY REPORT HAS A NARRATIVE DESCRIPTION THAT AUGMENTS AND SHOULD BE CONSIDERED PART OF THIS CHART, AND VISE-VERSA FOR THIS AND ALL MY CHARTS.**

Claim Text from '906 Patent	Wei94
906-1.a: A method for running an application program in a computer network environment, comprising:	Wei94 discloses an application program. *See, e.g.*, : [Wei94] discloses "viola applications" that "range from a simple clock to a World Wide Web hypermedia browser (ViolaWWW)" all of which were intended to be run as computer code physically embodied on a medium. ([Wei94] at 1.) Wei94 discloses a computer network environment. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) The World Wide Web was a distributed hypermedia environment.
906-1.b: providing at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;	Wei94 discloses a client workstation. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) As such, [Wei94] discloses a browser on a client workstation that retrieved documents from a World Wide Web server. Wei94 discloses a network server. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) As such, [Wei94] discloses a browser on a client workstation that retrieved documents from

Claim Text from '906 Patent	Wei94
	a World Wide Web server. Wei94 discloses a distributed hypermedia environment. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) The World Wide Web was a distributed hypermedia environment.
906-1.c: executing, at said client workstation, a browser application, that parses a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and for responding to predetermined text formats to initiate processing specified by said text formats;	Wei94 discloses a browser application. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) Wei94 discloses that the browser application parses a hypermedia document. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) These were markup languages which, as was well known in the art, were parsed by browsers. Wei94 discloses a hypermedia document with text formats. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) These documents were structured based on text formats.
906-1.d: utilizing said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server,	Wei94 discloses that a hypermedia document is received from the server. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) As such, [Wei94] discloses a browser on a client workstation that retrieved documents from a World Wide Web server.

Claim Text from '906 Patent	Wei94
	These documents were hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) Wei94 discloses that the browser displays a hypermedia document. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.)
906-1.e: wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation,	Wei94 discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, : [Wei94] includes various screenshots showing hypermedia documents displayed in a browser window. (See, e.g., [Wei94] at 5-10.)
906-1.f: wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document,	Wei94 discloses an embed text format at a first location in a hypermedia document. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format

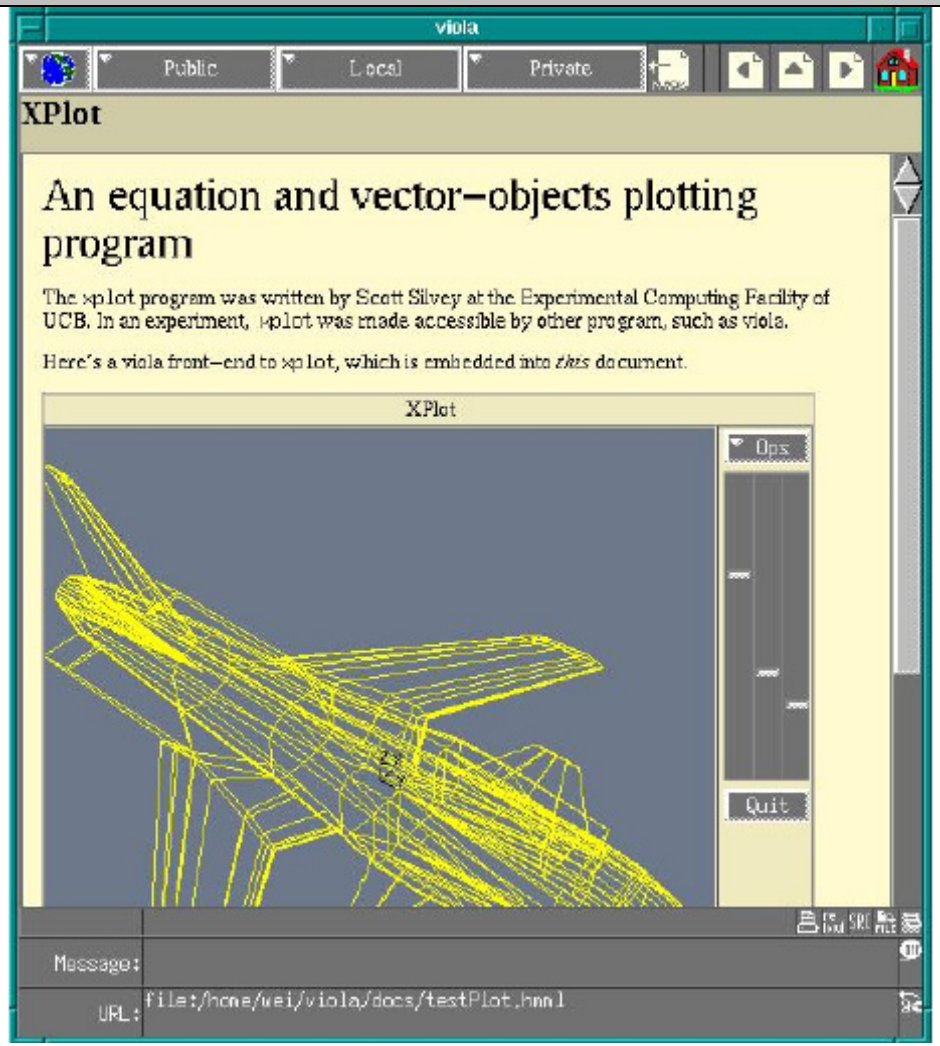
Claim Text from '906 Patent	Wei94
	called VOBJF, located at a first location in a hypermedia document. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.) Wei94 discloses that the embed text format specifies the location of an object. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.) The VOBJF embed text format specifies the location of an object. For example, testPlot.hmml includes a VOBJF tag that shows the tag's syntax, including that it specifies the location of an object based on a filepath location in which the object can be found: <VOBJF>/home/wei/viola/apps/plot.v<\VOBJF> (See viola\docs\testPlot.hmml.)

Claim Text from '906 Patent	Wei94
	Wei94 discloses an object that is external to a hypermedia document. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing application output. ([Wei94] at 7-10.) With reference to [Wei94] at 10, the jetfighter rendering is an object. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the graphing application, generated by the vplot application, has data for a default grid specified in the file plot.v by the command: output("equation 0"); (See apps\plot.v.)
906-1.g: wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document, and	Wei94 discloses that the object has associated type information. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that an embedded mini application can be achieved using an embed text format called VOBJF: <VOBJF>/home/wei/viola/apps/plot.v<\VOBJF> (See, e.g., viola\docs\testPlot.hmml) The file plot.v contains type information associated with the object. /path {/home/wei/vplot/vplot} (See viola\apps\plot.v.) Wei94 discloses that the browser uses type information to identify and locate an

Claim Text from '906 Patent	Wei94

	executable application. *See, e.g.*, :
	
	[Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.)
	ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent.
	Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that an embedded mini applications can be achieved using an embed text format called VOBJF:
	<VOBJF>/home/wei/viola/apps/plot.v<\VOBJF>
	(See, e.g., viola\docs\testPlot.hmml)
	The file plot.v contains type information associated with the object.
	/path {/home/wei/vplot/vplot}
	(See viola\apps\plot.v.)
	The type information is used by the ViolaWWW to identify and locate the vplot executable application. ViolaWWW then invokes the executable application.
	switch (pid = vfork()) {
	…
	case 0: * Child *\
	…
	execv(GET_path(self), args);
	(See viola\src\cl_TTY.c.)
	
	Wei94 discloses that the executable application is external to the hypermedia document. *See, e.g.*, :
	
	[Wei94] states that "[t]his next mini application front-ends a graphing process (on the same machine as the viola process). An important thing to

Claim Text from '906 Patent	Wei94
	note is that, like all the other document-embeddable mini applications shown, no special modification to the viola engine is required for ViolaWWW to support them. All the bindings are done via the viola language, provided that the necessary primitives are available in the interpreter, of course." In [www-talk-00293128], Pei Wei explained that this described Viola in operation with the vplot executable application, which was external to the hypermedia document: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." Further in [Wei94], Wei teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
906-1.h: wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable an end-user to directly interact with said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.	Wei94 discloses that the browser parses the embed text format. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.)

Claim Text from ’906 Patent	Wei94
	 Browser With Embedded App

Claim Text from '906 Patent	Wei94
	As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format," parsed by a browser, for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.) Wei94 discloses automatic invocation of the executable application. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graphing output. ([Wei94] at 10.) In [www-talk-00293128], Pei Wei explained that this described Viola in operation with the vplot executable application, which displayed graphing objects: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it."

Claim Text from ’906 Patent	Wei94

	Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that when ViolaWWW parses the VOBJF tag in testPlot.hmml, the vplot application is automatically invoked as follows:
	switch (pid = vfork()) {
	…
	case 0: * Child *\
	…
	execv(GET_path(self), args);
	(See src\cl_TTY.c.)
	
	Wei94 discloses that the executable application displays the object. *See, e.g.*, :
	
	[Wei94] states that "[t]his next mini application front-ends a graphing process (on the same machine as the viola process). An important thing to note is that, like all the other document-embeddable mini applications shown, no special modification to the viola engine is required for ViolaWWW to support them. All the bindings are done via the viola language, provided that the necessary primitives are available in the interpreter, of course."
	In [www-talk-00293128], Pei Wei explained that this described Viola in operation with the vplot executable application, which displayed graphing objects: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it."
	Further in [Wei94], Wei teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, “Here's another example of a mini interactive application that is embedded

Claim Text from '906 Patent	Wei94
	into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.] Wei94 discloses that the executable application enables direct interaction with the object. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graphing output. ([Wei94] at 10.) As shown in the figure depicting this example, there are slider bars that enable a user to interact directly with the jetfighter. Wei94 discloses that interaction with the object is at a first location in the hypermedia document. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graphing output. ([Wei94] at 10.) As shown in the figure depicting this example, there are slider bars that enable a user to interact directly with the jetfighter at the first location in the hypermedia document.
906-2.a: The method of claim 1, wherein said executable application is a controllable application and further comprising the step of: interactively controlling said controllable application on said client workstation via inter-process communications between said browser and said controllable	Wei94 discloses interactive control via inter-process communications between a browser and an application. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super

Claim Text from ’906 Patent	Wei94
application.	computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the “plotting demo” was produced. I have examined the relevant Viola code from [Viola-DX34] and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to [Viola-DX34] code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the [Viola-DX34] codeset and shows the resulting display. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, “Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more

Claim Text from '906 Patent	Wei94
	work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
906-3.a: The method of claim 2, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.	Wei94 discloses ongoing inter-process communications. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the "plotting demo" was produced. I have examined the relevant Viola code from [Viola-DX34] and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to [Viola-DX34] code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the [Viola-DX34] codeset and shows the resulting display.

Claim Text from '906 Patent	Wei94
	One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
906-6.a: A computer program product for use in a system having at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment, the computer program product comprising:	Wei94 discloses an application program in a computer network environment. *See* evidence recited for 906-1.a. Wei94 also discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 906-1.b.
906-6.b: a computer usable medium having computer readable program code physically embodied therein, said computer program product further comprising:	Wei94 discloses computer code physically embodied on a medium. *See, e.g.*, : The computer on which ViolaWWW executes includes computer usable media having computer readable program code physically embodied therein. [Wei94] discloses "viola applications" that "range from a simple clock to a World Wide Web hypermedia browser (ViolaWWW)," all of which were intended to be run as computer code physically embodied on a medium. ([Wei94] at 1.)
906-6.c: computer readable program code for causing said client workstation to execute a browser application to parse a first distributed hypermedia document to	Wei94 discloses a browser application that parses a hypermedia document with text formats. *See* evidence recited for 906-1.c.

Claim Text from '906 Patent	Wei94
identify text formats included in said distributed hypermedia document and to respond to predetermined text formats to initiate processes specified by said text formats;	
906-6.d: computer readable program code for causing said client workstation to utilize said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server,	Wei94 discloses a hypermedia document received from a server and a browser that displays the hypermedia document. *See* evidence recited for 906-1.d.
906-6.e: wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation,	Wei94 discloses that the hypermedia document is displayed in a browser window. *See* evidence recited for 906-1.e.
906-6.f: wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document,	Wei94 discloses an embed text format at a first location in a hypermedia document; that the embed text format specifies the location of an object; and that the object is external to the hypermedia document. *See* evidence recited for 906-1.f.
906-6.g: wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document, and	Wei94 discloses that the object has associated type information, that the browser uses the type information to identify and locate an executable application, and that the executable application is external to the hypermedia document. *See* evidence recited for 906-1.g.
906-6.h: wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable an end-user	Wei94 discloses that the browser parses the embed text format; that the browser automatically invokes the executable application; that the executable application displays the object and enables an end-user to directly interact with it; and that interaction with the object is at a first location in the hypermedia document. *See* evidence recited for 906-1.h.

Claim Text from '906 Patent	Wei94
to directly interact with said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.	
906-7.a: The computer program product of claim 6, wherein said executable application is a controllable application and further comprising: computer readable program code for causing said client workstation to interactively control said controllable application on said client workstation via inter-process communications between said browser and said controllable application.	Wei94 discloses interactive control via inter-process communications between a browser and an application. *See* evidence recited for 906-2.a.
906-8.a: The computer program product of claim 7, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.	Wei94 discloses ongoing inter-process communications. *See* evidence recited for 906-3.a.
906-11.a: The method of claim 3, wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes the following substeps:	Wei94 discloses additional instructions on the server. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple

Claim Text from ’906 Patent	Wei94
	protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the “plotting demo” was produced. I have examined the relevant Viola code from DX34 and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to DX34 code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the DX34 codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in

Claim Text from '906 Patent	Wei94
	the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
906-11.b: issuing, from the client workstation, one or more commands to the network server;	Wei94 discloses that the client issues commands to the server. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the "plotting demo" was produced. I have examined the relevant Viola code from DX34 and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to DX34 code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a

Claim Text from '906 Patent	Wei94
	VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the DX34 codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
906-11.c: executing, on the network server, one or more instructions in response to said commands;	Wei94 discloses that the server executes instructions in response to client commands. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it."

Claim Text from ’906 Patent	Wei94
	The preceding description by Wei is the essence of how the “plotting demo” was produced. I have examined the relevant Viola code from DX34 and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to DX34 code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the DX34 codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, “Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually

Claim Text from '906 Patent	Wei94
	active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
906-11.d: sending information from said network server to said client workstation in response to said executed instructions; and	Wei94 discloses that the server responds with information to the client. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the "plotting demo" was produced. I have examined the relevant Viola code from DX34 and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to DX34 code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v

Claim Text from '906 Patent	Wei94
	and testPlot.hmml from the DX34 codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
906-11.e: processing said information at the client workstation to interactively control said controllable application.	Wei94 discloses that the client uses information from the server to interactively control the application. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the "plotting demo" was produced. I have examined the relevant Viola code from DX34

Claim Text from ’906 Patent	Wei94
	and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to DX34 code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the DX34 codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, “Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more

Claim Text from '906 Patent	Wei94
	work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
906-13.a: The computer program product of claim 8, wherein additional instructions for controlling said controllable application reside on said network server, wherein said computer readable program code for causing said client workstation to interactively control said controllable application on said client workstation includes:	Wei94 discloses additional instructions on the server *See* evidence recited for 906-11.a.
906-13.b: computer readable program code for causing said client workstation to issue from the client workstation, one or more commands to the network server;	Wei94 discloses that the client issues commands to the server. *See* evidence recited for 906-11.b.
906-13.c: computer readable program code for causing said network server to execute one or more instructions in response to said commands;	Wei94 discloses that the server executes instructions in response to client commands. *See* evidence recited for 906-11.c.
906-13.d: computer readable program code for causing said network sever to send information to said client workstation in response to said executed instructions; and	Wei94 discloses that the server responds with information to the client. *See* evidence recited for 906-11.d.
906-13.e: computer readable program code for causing said client workstation to process said information at the client workstation to interactively control said controllable application.	Wei94 discloses that the client uses information from the server to interactively control the application. *See* evidence recited for 906-11.e..

INVALIDITY CLAIM CHART FOR U.S. PATENT NO. 7,599,985

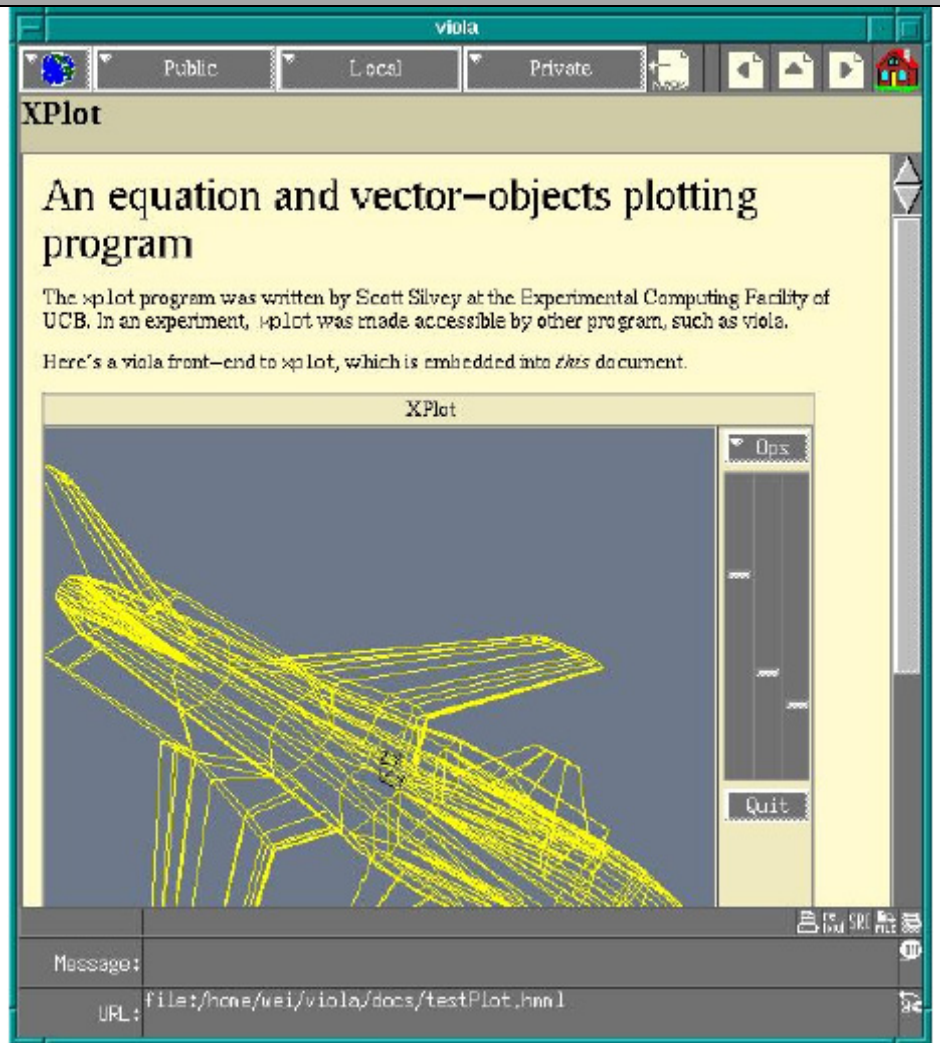
- **PEI WEI'S PUBLICATION "A BRIEF OVERVIEW OF THE VIOLA ENGINE, AND ITS APPLICATIONS" BEARING THE DATE AUGUST 16, 1994, AVAILABLE AT [PA-00318355]. ALSO AVAILABLE AT [PA-00318385] THROUGH [PA-00318392]. ("WEI94"). THE BODY OF MY REPORT HAS A NARRATIVE DESCRIPTION THAT AUGMENTS AND SHOULD BE CONSIDERED PART OF THIS CHART, AND VISE-VERSA FOR THIS AND ALL MY CHARTS.**

Claim Text from '985 Patent	Wei94
985-1.a: A method for running an application program in a distributed hypermedia network environment, wherein the network environment comprises at least one client workstation and one network server coupled to the network environment, the method comprising:	Wei94 discloses an application program. *See, e.g.*, : [Wei94] discloses "viola applications" that "range from a simple clock to a World Wide Web hypermedia browser (ViolaWWW)" all of which were intended to be run as computer code physically embodied on a medium. ([Wei94] at 1.) Wei94 discloses a computer network environment. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) The World Wide Web was a distributed hypermedia environment. Wei94 discloses a client workstation. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) As such, [Wei94] discloses a browser on a client workstation that retrieved documents from a World Wide Web server. Wei94 discloses a network server. *See, e.g.*, :

Claim Text from '985 Patent	Wei94
	[Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) As such, [Wei94] discloses a browser on a client workstation that retrieved documents from a World Wide Web server. Wei94 discloses a distributed hypermedia environment. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) The World Wide Web was a distributed hypermedia environment.
985-1.b: receiving, at the client workstation from the network server over the network environment, at least one file containing information to enable a browser application to display at least a portion of a distributed hypermedia document within a browser-controlled window;	Wei94 discloses a browser application. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) Wei94 discloses a file containing enabling information. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) These were examples of files containing enabling information. Wei94 discloses that the file is received at the client workstation from the network server. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.) As such, [Wei94] discloses a browser on a client workstation that retrieved files from a World Wide Web server. These documents were hypermedia files, including HTML files ([Wei94] at 2); and HMML files ([Wei94] at 4.) Wei94 discloses that the browser displays at least a portion of a distributed

Claim Text from '985 Patent	Wei94
	hypermedia document. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) Wei94 discloses that at least a portion of a hypermedia document is displayed in a browser-controlled window. *See, e.g.*, : [Wei94] includes various screenshots showing hypermedia documents displayed in a browser window. (See, e.g., [Wei94] at 5-10.)
985-1.c: executing the browser application on the client workstation, with the browser application:	Wei94 discloses a browser application executing on the client workstation. *See, e.g.*, : [Wei94] discloses "viola applications" that include "a World Wide Web hypermedia browser (ViolaWWW)." ([Wei94] at 1.)
985-1.d: responding to text formats to initiate processing specified by the text formats;	Wei94 discloses responding to text formats to initiate processing specified by the text formats, i.e., parsing text formats. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) As was well known in the art, these are examples of markup languages, which included text formats that were parsed by browsers.
985-1.e: displaying at least a portion of the document within the browser-controlled window;	Wei94 discloses that the browser displays a hypermedia document. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) Wei94 discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, : [Wei94] includes various screenshots showing hypermedia documents

Claim Text from '985 Patent	Wei94
	displayed in a browser window. (See, e.g., [Wei94] at 5-10.)
985-1.f: identifying an embed text format which corresponds to a first location in the document, where the embed text format specifies the location of at least a portion of an object external to the file, where the object has type information associated with it;	Wei94 discloses identifying an embed text format. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.)

Claim Text from '985 Patent	Wei94
	 Browser With Embedded App As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about

Claim Text from '985 Patent	Wei94
	embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.) HMML was a markup language, and markup language tags were identified by a browser, as was well known in the art. Wei94 discloses that the embed text format corresponds to a first location in the hypermedia document. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML

Claim Text from '985 Patent	Wei94
	markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.) The VOBJF text format corresponds to a first location in the hypermedia document. Wei94 discloses that the embed text format specifies the location of an object. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.)

Claim Text from '985 Patent	Wei94
	The VOBJF embed text format specifies the location of an object. For example, testPlot.hmml includes a VOBJF tag that shows the tag's syntax, including that it specifies the location of an object based on a filepath location in which the object can be found: <VOBJF>/home/wei/viola/apps/plot.v<\VOBJF> (See viola\docs\testPlot.hmml.) Wei94 discloses that the object is external to the file containing enabling information. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing application output. ([Wei94] at 7-10.) With reference to [Wei94] at 10, the jetfighter rendering is an object. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the graphing application, generated by the vplot application, has data for a default grid specified in the file plot.v by the command: output("equation 0"); (See apps\plot.v.) Wei94 discloses that the object has associated type information. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that an

Claim Text from '985 Patent	Wei94
	embedded mini applications can be achieved using an embed text format called VOBJF: <VOBJF>/home/wei/viola/apps/plot.v<\VOBJF> (See, e.g., viola\docs\testPlot.hmml) The file plot.v contains type information associated with the object. /path {/home/wei/vplot/vplot} (See viola\apps\plot.v.)
985-1.g: utilizing the type information to identify and locate an executable application external to the file; and	Wei94 discloses that the browser uses type information to identify and locate an executable application. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that an embedded mini applications can be achieved using an embed text format called VOBJF: <VOBJF>/home/wei/viola/apps/plot.v<\VOBJF> (See, e.g., viola\docs\testPlot.hmml) The file plot.v contains type information associated with the object. /path {/home/wei/vplot/vplot} (See viola\apps\plot.v.) The type information is used by the ViolaWWW to identify and locate the vplot executable application. ViolaWWW then invokes the executable application. switch (pid = vfork()) { … case 0: * Child *\ …

Claim Text from '985 Patent	Wei94
	execv(GET_path(self), args); (See viola\src\cl_TTY.c.) Wei94 discloses that the executable application is external to the file containing enabling information. *See, e.g.*, : [Wei94] states that "[t]his next mini application front-ends a graphing process (on the same machine as the viola process). An important thing to note is that, like all the other document-embeddable mini applications shown, no special modification to the viola engine is required for ViolaWWW to support them. All the bindings are done via the viola language, provided that the necessary primitives are available in the interpreter, of course." In [www-talk-00293128], Pei Wei explained that this described Viola in operation with the vplot executable application, which was external to the hypermedia document: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." Further in [Wei94], Wei teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
985-1.h: automatically invoking the executable application,	Wei94 discloses that the browser parses the embed text format. *See, e.g.*, :

Claim Text from '985 Patent	Wei94
in response to the identifying of the embed text format, to execute on the client workstation in order to display the object and enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of the hypermedia document being displayed in the browser-controlled window.	[Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format," parsed by a browser, for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.) Wei94 discloses automatic invocation of the executable application. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graphing output. ([Wei94] at 10.) In [www-talk-00293128], Pei Wei explained that this described Viola in operation with the vplot executable application, which displayed graphing objects: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-

Claim Text from '985 Patent	Wei94

	end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it."
	Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that when ViolaWWW parses the VOBJF tag in testPlot.hmml, the vplot application is automatically invoked as follows:
	switch (pid = vfork()) {
	…
	case 0: * Child *\
	…
	execv(GET_path(self), args);
	(See src\cl_TTY.c.)
	
	Wei94 discloses that the executable application displays the object. *See, e.g.*, :
	
	[Wei94] states that "[t]his next mini application front-ends a graphing process (on the same machine as the viola process). An important thing to note is that, like all the other document-embeddable mini applications shown, no special modification to the viola engine is required for ViolaWWW to support them. All the bindings are done via the viola language, provided that the necessary primitives are available in the interpreter, of course."
	In [www-talk-00293128], Pei Wei explained that this described Viola in operation with the vplot executable application, which displayed graphing objects: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it."

Claim Text from '985 Patent	Wei94
	Further in [Wei94], Wei teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.] Wei94 discloses that the executable application enables direct interaction with the object. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graphing output. ([Wei94] at 10.) As shown in the figure depicting this example, there are slider bars that enable a user to interact directly with the jetfighter. Wei94 discloses that interaction with the object is at a first location in the hypermedia document. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graphing output. ([Wei94] at 10.) As shown in the figure depicting this example, there are slider bars that enable a user to interact directly with the jetfighter at the first location in the hypermedia document.
985-2.a: The method of claim 1 where: the information to enable comprises text formats.	Wei94 discloses that the enabling information in the file is text formats. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML

Claim Text from '985 Patent	Wei94
	documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) These were examples of files containing enabling information. The enabling information took the form of markup language text formats.
985-3.a: The method of claim 2 where the text formats are HTML tags.	Wei94 discloses that the text formats are HTML tags. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents. ([Wei94] at 2.)
985-4.a: The method of claim 1 where the information contained in the file received comprises at least one embed text format.	Wei94 discloses that the enabling information in the file includes an embed text format. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.)

Claim Text from '985 Patent	Wei94
985-5.a: The method of claim 1 where the step of identifying an embed text format comprises: parsing the received file to identify text formats included in the received file.	Wei94 discloses that the embed text format is identified by parsing the file containing enabling information. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.) HMML was a markup language, and markup language tags were parsed by a browser, as was well known in the art.
985-6.a: The method of claim 5 where the parsing is by a parser in the browser.	Wei94 discloses that the parser is in the browser *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) As was well known in the art, these were examples of files with markup languages that were processed by browsers with parsers.

Claim Text from '985 Patent	Wei94
985-7.a: The method of claim 1 where the processing specified by the text formats is specified directly.	Wei94 discloses that the text formats directly specify the processing. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) It was well known in the art that these types of documents included text formats that directly specified processing. By way of example, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. It includes an exemplary HMML file (testPlot.hmml) which contains the HMML tags such as TITLE, H1 and ITALIC. It also includes an exemplar HTML file (testAll.html) which contains HTML tags, such as TITLE and H1. The hypermedia document downloaded from the remote network server would be parsed by ViolaWWW to identify these tags. ViolaWWW then initiates processing directly specified by the tags. For example, ViolaWWW displays the text marked by the H1 tag in large, bold, header text and the text marked by the ITALIC tag in italics.
985-8.a: The method of claim 1 where the correspondence is implied by the order of the text format in a set of all of the text formats.	Wei94 discloses that the correspondence is implied by the order of text formats. *See, e.g.*, : ViolaWWW displayed hypermedia documents, including HTML documents ([Wei94] at 2); and HMML documents ([Wei94] at 4.) With HTML and HMML, the correspondence between the location in the document and the text formats is implied by the order of the text formats. For example, with reference to [Viola-DX34] as an example, in testPlot.hmml, TITLE tag appears before H1 tag. H1 tag is followed by P tag. VOBJF tag appears later. When ViolaWWW displays the document, the title (associated with TITLE tag) is displayed first. The title is followed by heading (associated with H1 tag). A paragraph (associated with P tag) is displayed after the heading. An object is embedded later in the document where VOBJF tag is specified. Similarly, again using [Viola-DX34] as an example, for testAll.html, a

Claim Text from '985 Patent	Wei94
	title (associated with a TITLE tag) is displayed ahead of Header 1 (associated with a subsequent header tag), which is displayed ahead of Header 2 (associated with a still subsequent header tag). (See docs\testAll.html.)
985-9.a: The method of claim 1 where the embed text format specifies the location of at least a portion of an object directly.	Wei94 discloses that the embed text format specifies the location of the object directly. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) As to the graphing output, it is clear that the depiction of the jetfighter is "embedded" for at least three reasons. First, the paper talks about embedding and discusses this example in connection with "embedding mini applications." Second, the image of the jetfighter does not include the control adornments that one typically associates with standalone windows in the X Windows environment. Third, Pei Wei told Michael Doyle that this plotting demo was an embedded object, at least in [www-talk-00293126]. ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that the embedded mini applications can be achieved using an embed text format called VOBJF. (See, e.g., viola\docs\testPlot.hmml; docs\violaChier.hmml.) The VOBJF embed text format specifies the location of an object directly. For example, testPlot.hmml includes a VOBJF tag specifying the location of an object based on a filepath location in which the object can be found: <VOBJF>/home/wei/viola/apps/plot.v<\VOBJF> (See viola\docs\testPlot.hmml.)

Claim Text from '985 Patent	Wei94
985-10.a: The method of claim 1 where having type information associated is by including type information in the embed text format.	Wei94 discloses that the type information is in the embed text format. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing output. ([Wei94] at 7-10.) ViolaWWW structured documents according to HTML and HMML markup languages. ([Wei94] at 2; 4.) Accordingly, the use of an "embed text format" for embedding these mini applications is inherent. Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that an embedded mini applications can be achieved using an embed text format called VOBJF: <VOBJF>/home/wei/viola/apps/plot.v<\VOBJF> (See, e.g., viola\docs\testPlot.hmml) The file plot.v contains type information associated with the object. The file plot.v (which contains type information as described above) is in the VOBJF embed text format.
985-11.a: The method of claim 1 where automatically invoking does not require interactive action by the user.	Wei94 discloses that automatic invocation does not require interactive action by the user. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graphing output. ([Wei94] at 10.) In [www-talk-00293128], Pei Wei explained that this described Viola in operation with the vplot executable application, which displayed graphing objects: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end

Claim Text from '985 Patent	Wei94

	draws the graphics directly onto the window violaWWW has opened for it."
	Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. This shows that when ViolaWWW parses the VOBJF tag in testPlot.hmml, the vplot application is automatically invoked, without requiring interactive action, as follows:
	switch (pid = vfork()) {
	…
	case 0: * Child *\
	…
	execv(GET_path(self), args);
	(See src\cl_TTY.c.)

985-16.a: One or more computer readable media encoded with software comprising computer executable instructions, for use in a distributed hypermedia network environment, wherein the network environment comprises at least one client workstation and one network server coupled to the network environment, and when the software is executed operable to:	Wei94 discloses computer code physically embodied on a medium. *See, e.g.*, : The computer on which ViolaWWW executes includes computer usable media having computer readable program code physically embodied therein. [Wei94] discloses "viola applications" that "range from a simple clock to a World Wide Web hypermedia browser (ViolaWWW)," all of which were intended to be run as computer code physically embodied on a medium. ([Wei94] at 1.) Wei94 discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-16.b: receive, at the client workstation from the network server over the network environment, at least one file containing information to enable a browser application to display at least a portion of a distributed hypermedia document within a browser-controlled window;	Wei94 discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.

Claim Text from '985 Patent	Wei94
985-16.c: cause the client workstation to utilize the browser to:	Wei94 discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.
985-16.d: respond to text formats to initiate processing specified by the text formats;	Wei94 discloses parsing text formats. *See* evidence recited for 985-1.d.
985-16.e: display at least a portion of the document within the browser-controlled window;	Wei94 discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-16.f: identify an embed text format corresponding to a first location in the document, the embed text format specifying the location of at least a portion of an object external to the file, with the object having type information associated with it;	Wei94 discloses identifying an embed text format; that the embed text format corresponds to a first location in a hypermedia document; that the embed text format specifies the location of at least a portion of an object external to the file containing enabling information; and that the object has associated type information. *See* evidence recited for 985-1.f.
985-16.g: utilize the type information to identify and locate an executable application external to the file; and	Wei94 discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.
985-16.h: automatically invoke the executable application, in response to the identifying of the embed text format, to execute on the client workstation in order to display the object and enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of the hypermedia document being displayed in the browser-controlled window.	Wei94 discloses automatically invoking the executable application; that the executable application displays the object and enables an end-user to directly interact with it; and that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h.
985-17.a: The computer readable media of claim 16 where: the information to enable comprises text formats.	Wei94 discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.

Claim Text from '985 Patent	Wei94
985-18.a: The computer readable media of claim 17 where: the text formats are HTML tags.	Wei94 discloses that the text formats are HTML tags. *See* evidence recited for 985-3.a.
985-19.a: The computer readable media of claim 16 where: the information contained in the file received comprises at least one embed text format.	Wei94 discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.
985-20.a: A method of serving digital information in a computer network environment having a network server coupled the network environment, and where the network environment is a distributed hypermedia environment, the method comprising:	Wei94 discloses digital information. *See, e.g.*, : The information that is exchanged between a client workstation running ViolaWWW and a server is digital information. For example, ViolaWWW running on the client workstation can receive hypermedia HTML and HMML documents from a network server over the World Wide Web. ([Wei94] at 1; 2; 4.) These documents would be transmitted according to network protocols that transmit information in digital form. Wei94 discloses a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-20.b: communicating via the network server with at least one client workstation over said network in order to cause said client workstation to:	Wei94 discloses a client workstation. *See* evidence recited for 985-1.a. Wei94 discloses communicating via network server in order to cause the client workstation to act. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple

Claim Text from ’985 Patent	Wei94
	protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the “plotting demo” was produced. I have examined the relevant Viola code from DX34 and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to DX34 code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the DX34 codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, “Here's

Claim Text from '985 Patent	Wei94
	another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
985-20.c: receive, over said network environment from said server, at least one file containing information to enable a browser application to display at least a portion of a distributed hypermedia document within a browser-controlled window;	Wei94 discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.
985-20.d: execute, at said client workstation, a browser application, with the browser application:	Wei94 discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.
985-20.e: responding to text formats to initiate processing specified by the text formats;	Wei94 discloses parsing text formats. *See* evidence recited for 985-1.d.
985-20.f: displaying, on said client workstation, at least a portion of the document within the browser-controlled window;	Wei94 discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-20.g: identifying an embed text format which corresponds to a first location in the document, where the embed text format specifies the location of at least a portion of an object external to the file, where the object has type information associated with it;	Wei94 discloses identifying an embed text format; that the embed text format corresponds to a first location in a hypermedia document; that the embed text format specifies the location of at least a portion of an object external to the file containing enabling information; and that the object has associated type information. *See* evidence recited for 985-1.f.
985-20.h: utilizing the type information to identify and locate an executable application external to the file; and	Wei94 discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.

Claim Text from '985 Patent	Wei94
985-20.i: automatically invoking the executable application, in response to the identifying of the embed text format, to execute on the client workstation in order to display the object and enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of the hypermedia document being displayed in the browser-controlled window.	Wei94 discloses automatically invoking the executable application; that the executable application displays the object and enables an end-user to directly interact with it; and that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h.
985-21.a: The method of claim 20 where: the information to enable comprises text formats.	Wei94 discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-22.a: The method of claim 21 where: the text formats are HTML tags.	Wei94 discloses that the text formats are HTML tags. *See* evidence recited for 985-3.a.
985-23.a: The method of claim 20 where: the information contained in the file received comprises at least one embed text format.	Wei94 discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.
985-24.a: A method for running an executable application in a computer network environment, wherein said network environment has at least one client workstation and one network server coupled to a network environment, the method comprising:	Wei94 discloses a client workstation and a network server in a network environment. *See* evidence recited for 985-1.a. Wei94 discloses an executable application. *See* evidence recited for 985-1.g.
985-24.b: enabling an end-user to directly interact with an	Wei94 discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.

Claim Text from '985 Patent	Wei94
object by utilizing said executable application to interactively process said object while the object is being displayed within a display area created at a first location within a portion of a hypermedia document being displayed in a browser-controlled window,	Wei94 discloses an object external to a file containing enabling information. *See* evidence recited for 985-1.f. Wei94 discloses that there is enabling of an end-user to directly interact with the object. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graphing output. ([Wei94] at 10.) As shown in the figure depicting this example, there are slider bars that enable a user to interact directly with the jetfighter. Wei94 discloses that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h. Wei94 discloses that the object is displayed at a first location within a portion of the hypermedia document being displayed. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing application output. ([Wei94] at 7-10.) Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. With reference to the graphing application, the vplot application displays the object as a grid (the default grid) inside the ViolaWWW window. The object is displayed at the first location in the portion of the testPlot.hmml hypermedia document being displayed in the ViolaWWW window. (See, e.g., viola\docs\testPlot.hmml.)
985-24.c: wherein said network environment is a distributed	Wei94 discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.

Claim Text from '985 Patent	Wei94
hypermedia environment,	
985-24.d: wherein said client workstation receives, over said network environment from said server, at least one file containing information to enable said browser application to display, on said client workstation, at least said portion of said distributed hypermedia document within said browser-controlled window,	Wei94 discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.
985-24.e: wherein said executable application is external to said file,	Wei94 discloses an executable application external to the file. *See* evidence recited for 985-1.g.
985-24.f: wherein said client workstation executes the browser application, with the browser application responding to text formats to initiate processing specified by the text formats,	Wei94 discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c. Wei94 discloses parsing text formats. *See* evidence recited for 985-1.d.
985-24.g: wherein at least said portion of the document is displayed within the browser-controlled window,	Wei94 discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-24.h: wherein an embed text format which corresponds to said first location in the document is identified by the browser,	Wei94 discloses identifying an embed text format and that the embed text format corresponds to a first location in a hypermedia document. *See* evidence recited for 985-1.f.
985-24.i: wherein the embed text format specifies the location of at least a portion of said object external to the file,	Wei94 discloses that the embed text format specifies the location of at least a portion of an object external to the file containing enabling information. *See* evidence recited for 985-1.f.
985-24.j: wherein the object has type information associated with it,	Wei94 discloses that the object has associated type information. *See* evidence recited for 985-1.f.
985-24.k: wherein the type information is utilized by the	Wei94 discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.

Claim Text from '985 Patent	Wei94
browser to identify and locate said executable application, and	
985-24.l: wherein the executable application is automatically invoked by the browser, in response to the identifying of the embed text format.	Wei94 discloses automatically invoking the executable application. *See* evidence recited for 985-1.h.
985-25.a: The method of claim 24 where: the information to enable comprises text formats.	Wei94 discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-26.a: The method of claim 25 where: the text formats are HTML tags.	Wei94 discloses that the text formats are HTML tags. *See* evidence recited for 985-3.a.
985-27.a: The method of claim 24 where: the information contained in the file received comprises at least one embed text format.	Wei94 discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.
985-28.a: One or more computer readable media encoded with software comprising an executable application for use in a system having at least one client workstation and one network server coupled to a network environment, operable to:	Wei94 discloses computer code physically embodied on a medium. *See* evidence recited for 985-16.a. Wei94 discloses a client workstation and a network server in a network environment. *See* evidence recited for 985-1.a. Wei94 discloses an executable application. *See* evidence recited for 985-1.g.
985-28.b: cause the client workstation to display an object and enable an end-user to directly interact with said object while the object is being displayed within a display area created at a first location	Wei94 discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e. Wei94 discloses an object external to a file containing enabling information. *See* evidence recited for 985-1.f.

Claim Text from '985 Patent	Wei94
within a portion of a hypermedia document being displayed in a browser-controlled window,	Wei94 discloses that there is enabling of an end-user to directly interact with the object. *See* evidence recited for 985-24.b. Wei94 discloses that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h. Wei94 discloses that the object is displayed within a display area created at the first location.. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing application output. ([Wei94] at 7-10.) Also, in [www-talk-00293126], Pei Wei references his demonstration of [Viola-DX34]; a codebase which I analyzed. With reference to the graphing application, the vplot application displays the object as a grid (the default grid) inside the ViolaWWW window. The object is displayed at the first location in the portion of the testPlot.hmml hypermedia document being displayed in the ViolaWWW window. (See, e.g., viola\docs\testPlot.hmml.)
985-28.c: wherein said network environment is a distributed hypermedia environment,	Wei94 discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-28.d: wherein said client workstation receives, over said network environment from said server, at least one file containing information to enable said browser application to display, on said client workstation, at least said portion of said distributed hypermedia document within said browser-controlled window,	Wei94 discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.
985-28.e:	Wei94 discloses an executable application external to the file. *See* evidence

Claim Text from '985 Patent	Wei94
wherein said executable application is external to said file,	recited for 985-1.g.
985-28.f: wherein said client workstation executes said browser application, with the browser application responding to text formats to initiate processing specified by the text formats,	Wei94 discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c. Wei94 discloses parsing text formats. *See* evidence recited for 985-1.d.
985-28.g: wherein at least said portion of the document is displayed within the browser-controlled window,	Wei94 discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-28.h: wherein an embed text format which corresponds to said first location in the document is identified by the browser,	Wei94 discloses identifying an embed text format and that the embed text format corresponds to a first location in a hypermedia document. *See* evidence recited for 985-1.f.
985-28.i: wherein the embed text format specifies the location of at least a portion of said object external to the file,	Wei94 discloses that the embed text format specifies the location of at least a portion of an object external to the file containing enabling information. *See* evidence recited for 985-1.f.
985-28.j: wherein the object has type information associated with it,	Wei94 discloses that the object has associated type information. *See* evidence recited for 985-1.f.
985-28.k: wherein the type information is utilized by the browser to identify and locate said executable application, and	Wei94 discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.
985-28.l: wherein the executable application is automatically invoked by the browser, in response to the identifying of the embed text format.	Wei94 discloses automatically invoking the executable application. *See* evidence recited for 985-1.h.
985-36.a: A method for running an application program in a	Wei94 discloses an application program in a distributed hypermedia environment comprising at least client workstation and network server. *See*

Claim Text from '985 Patent	Wei94
distributed hypermedia network environment, wherein the distributed hypermedia network environment comprises at least one client workstation and one remote network server coupled to the distributed hypermedia network environment, the method comprising:	evidence recited for 985-1.a.
985-36.b: receiving, at the client workstation from the network server over the distributed hypermedia network environment, at least one file containing information to enable a browser application to display at least a portion of a distributed hypermedia document within a browser-controlled window;	Wei94 discloses a browser application; a file containing enabling information; that the file is received at the client workstation from the network server; that the browser displays at least a portion of a distributed hypermedia document; and that at least a portion of a hypermedia document is displayed in a browser-controlled window. *See* evidence recited for 985-1.b.
985-36.c: executing the browser application on the client workstation, with the browser application:	Wei94 discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.
985-36.d: responding to text formats to initiate processing specified by the text formats;	Wei94 discloses parsing text formats. *See* evidence recited for 985-1.d.
985-36.e: displaying at least a portion of the document within the browser-controlled window;	Wei94 discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-36.f: identifying an embed text format which corresponds to a first location in the document, where the embed text format specifies the location of at least a portion of an object;	Wei94 discloses an object. *See, e.g.*, : [Wei94] includes a section entitled "Embedding mini applications," which gives examples of embedded objects including a graph object, a chess board, a message relay, and graphing application output. ([Wei94] at 7-10.) With reference to [Wei94] at 10, the jetfighter rendering is an object. Wei94 discloses identifying an embed text format; that the embed text format corresponds to a first location in the hypermedia document; and that the embed

Claim Text from '985 Patent	Wei94
	text format specifies the location of an object. *See* evidence recited for 985-1.f.
985-36.g: identifying and locating an executable application associated with the object; and	Wei94 discloses that the browser identifies and locates an executable application associated with the object. *See, e.g.*, [Wei94] states that "[t]his next mini application front-ends a graphing process (on the same machine as the viola process). An important thing to note is that, like all the other document-embeddable mini applications shown, no special modification to the viola engine is required for ViolaWWW to support them. All the bindings are done via the viola language, provided that the necessary primitives are available in the interpreter, of course." In [www-talk-00293128], Pei Wei explained that this described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." Further in [Wei94], Wei teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
985-36.h: automatically invoking the executable application, in response to the identifying of the embed text format, in order to enable an end-user to directly interact with the object, while the object is being	Wei94 discloses identifying an embed text format. *See* evidence recited in 985-1.f. Wei94 discloses automatic invocation of the executable application; that the executable application displays the object; that the executable application

Claim Text from '985 Patent	Wei94
displayed within a display area created at the first location within the portion of the hypermedia document being displayed in the browser-controlled window,	enables direct interaction with the object; and that interaction with the object is at a first location in the hypermedia document. *See* evidence recited in 985-1.h. Wei94 discloses that the object is displayed at a first location within a portion of the hypermedia document being displayed. *See* evidence recited at 985-24.b. Wei94 discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, evidence recited for 985-1.e.
985-36.i: wherein the executable application is part of a distributed application, and	Wei94 discloses a distributed application. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the "plotting demo" was produced. I have examined the relevant Viola code from [Viola-DX34] and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei suggested ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi],

Claim Text from ’985 Patent	Wei94
	that demonstrates how simple it is to make trivial changes to [Viola-DX34] code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the [Viola-DX34] codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, “Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola.” [Wei94 at 7.] Wei94 discloses that the executable application is part of a distributed application. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the

Claim Text from ’985 Patent	Wei94
	point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the “plotting demo” was produced. I have examined the relevant Viola code from DX34 and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to DX34 code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the DX34 codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described

Claim Text from '985 Patent	Wei94
	in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
985-36.j: wherein at least a portion of the distributed application is for execution on a remote network server coupled to the distributed hypermedia network environment.	Wei94 discloses that the distributed application executes at least partially on a network server. *See, e.g.*, : In [www-talk-00293128], Pei Wei described Viola in operation with the vplot executable application: "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." The preceding description by Wei is the essence of how the "plotting demo" was produced. I have examined the relevant Viola code from DX34 and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as Wei described ("the back-end could very well be running on a remote super computer") and as I will demonstrate in what follows. I have recently produced a video recording, [Viola video 9.avi], that demonstrates how simple it is to make trivial changes to DX34 code to effect a situation where ViolaWWW runs on a client workstation and

Claim Text from '985 Patent	Wei94
	accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the DX34 codeset and shows the resulting display. It is important to recognize that Wei told Doyle exactly how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has passed VIS an X window ID which it uses to draw the graphics directly onto the window violaWWW has opened for it. One other gem that Wei passed on to Doyle is that, in general, ViolaWWW can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the ViolaWWW browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94 at 7.]
985-37.a: The method of claim 36 where: the information to enable comprises text formats.	Wei94 discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-38.a: The method of claim 37 where: the text formats are HTML tags.	Wei94 discloses that the text formats are HTML tags. *See* evidence recited for 985-3.a.
985-39.a:	Wei94 discloses that the enabling information in the file includes an embed text

Claim Text from '985 Patent	Wei94
The method of claim 36 where: the information contained in the file received comprises at least one embed text format.	format. *See* evidence recited for 985-4.a.
985-40.a: A method of serving digital information in a computer network environment having a network server coupled to said computer network environment, and where the network environment is a distributed hypermedia network environment, the method comprising:	Wei94 discloses digital information. *See* evidence recited for 985-20.a. Wei94 discloses a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-40.b: communicating via the network server with at least one remote client workstation over said computer network environment in order to cause said client workstation to:	Wei94 discloses a client workstation. *See* evidence recited for 985-1.a. Wei94 discloses communicating via network server in order to cause the client workstation to act. *See* evidence recited for 985-20.b.
985-40.c: receive, over said computer network environment from the network server, at least one file containing information to enable a browser application to display at least a portion of a distributed hypermedia document within a browser-controlled window;	Wei94 discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.
985-40.d: execute, at said client workstation, a browser application, with the browser application:	Wei94 discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.
985-40.e: responding to text formats to initiate processing specified by the text formats;	Wei94 discloses parsing text formats. *See* evidence recited for 985-1.d.
985-40.f: displaying, on said client workstation, at least a portion of the document within the browser-	Wei94 discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.

Claim Text from '985 Patent	Wei94
controlled window;	
985-40.g: identifying an embed text format which corresponds to a first location in the document, where the embed text format specifies the location of at least a portion of an object;	Wei94 discloses an object. *See* evidence recited for 985-36.f. Wei94 discloses identifying an embed text format; that the embed text format corresponds to a first location in the hypermedia document; and that the embed text format specifies the location of an object. *See* evidence recited for 985-1.f.
985-40.h: identifying and locating an executable application associated with the object; and	Wei94 discloses that the browser identifies and locates an executable application associated with the object. *See* evidence recited for 985-36.g.
985-40.i: automatically invoking the executable application, in response to the identifying of the embed text format, in order to enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of the hypermedia document being displayed in the browser-controlled window,	Wei94 discloses identifying an embed text format. *See* evidence recited in 985-1.f. Wei94 discloses automatic invocation of the executable application; that the executable application displays the object; that the executable application enables direct interaction with the object; and that interaction with the object is at a first location in the hypermedia document. *See* evidence recited in 985-1.h. Wei94 discloses that the object is displayed at a first location within a portion of the hypermedia document being displayed. *See* evidence recited for 985-24.b. Wei94 discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, evidence recited for 985-1.e.
985-40.j: wherein the executable application is part of a distributed application, and	Wei94 discloses that the executable application is part of a distributed application. *See* evidence recited in 985-36.i.
985-40.k: wherein at least a portion of the distributed application is for execution on the network server.	Wei94 discloses that the distributed application executes at least partially on a network server. *See* evidence recited for 985-36.j.

Claim Text from '985 Patent	Wei94
985-41.a: The method of claim 40 where: the information to enable comprises text formats.	Wei94 discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-42.a: The method of claim 41 where: the text formats are HTML tags.	Wei94 discloses that the text formats are HTML tags. *See* evidence recited for 985-3.a.
985-43.a: The method of claim 40 where: the information contained in the file received comprises at least one embed text format.	Wei94 discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.