Exhibit U

#15 pl 5-13

Application of: Doyle et al.

Application Num.: 90/006,831

Filed: October 30, 2003

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

Examiner: Caldwell, A.T.

Art unit: 2157

Declaration of Edward W. Felten

I, Edward W. Felten, declare as follows:

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

I. Qualifications

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

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

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

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

II. Relevant State of the Art in 1994

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

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

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

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

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

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

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

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

A. The Level of Skill In the Art

15. My benchmark for what ordinary skill in the art means is a person who is just graduating from a good computer science program at a college or a university – not a star student but just an average student – or a person who has gained an equivalent level of knowledge through experience in the industry. This person knows how to do things in conventional ways but does not exhibit an unusual level of innovative thinking.

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

B. The Grounds of Rejection

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

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

B. The '906 Patent

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

C. Prior Art Browsers

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

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

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

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

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

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

D. The Berners-Lee Reference

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

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

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

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

D. The Raggett I Reference

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

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

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

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

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

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

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

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

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

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

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

(Raggett I at p. 6)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

D. The Raggett II Reference

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

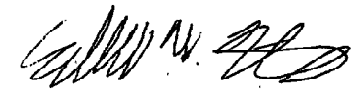
62. Finally, I have been told by the patent attorney for Eolas and the Regents that I should consider as part of my obviousness analysis "secondary considerations" such as copying, long felt but unresolved need, properties of the claimed invention, licenses showing industry acceptance of the invention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

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

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

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

Dated: May 7, 2004



Edward W. Felten