

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CARVANA, LLC
Petitioner,

v.

INTERNATIONAL BUSINESS MACHINES CORPORATION
Patent Owner.

Inter Partes Review No.: IPR2025-00564

U.S. Patent No. 7,702,719

**DECLARATION OF JOHN K. BENNETT, PH.D., P.E., IN SUPPORT OF
PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 7,702,719**

TABLE OF CONTENTS

	<u>Page</u>
I. Engagement	1
II. Qualifications.....	1
III. Materials Reviewed	4
IV. Legal Standards	4
A. Claim Construction	5
B. Anticipation.....	7
C. Obviousness	7
D. Level of Skill in the Art	9
V. Background of the Technology	10
A. Model-View-Controller (MVC) and “Dual-MVC” Were Well Known Years Before the ’719 Patent Was Filed	10
B. JavaScript Was Known at Least as of 1995.....	13
VI. The ’719 Patent Claims the Well-Known “Dual-MVC” Paradigm	14
A. Overview	14
B. ’719 Patent Claims	18
VII. Specific Examples of Prior Art Disclosures	23
A. U.S. Patent No. 5,838,909 (Roy)	23
(1) Roy Is Analogous Prior Art to the ’719 Patent	23
(2) Roy Discloses a “Dual-MVC” Networked System.....	24
B. U.S. Patent No. 5,695,400 (Fennell).....	28
(1) Fennell Is Analogous Prior Art to the ’719 Patent	28
(2) Fennell Is Related to Roy and Discloses Distributed Internet Applications Accessed Via Browser Software	30
C. JavaScript: The Definitive Guide (“O’Reilly”)	32

(1)	O'Reilly, a Printed Publication and Analogous Prior Art to the '719 Patent, Discloses the Use of Frames, Invisible Frames, HTML, JavaScript, and APIs Years before the '719 Patent Was Filed.....	32
(2)	O'Reilly Demonstrates a POSITA's Familiarity with HTTP as of at Least 1997.....	34
(3)	O'Reilly Discloses Using Frames and Invisible Frames to Generate HTML Documents and Store View-Generating and Controller Logic, the Same Subject Matter Claimed to Be (But Was Not) Invented in the '719 Patent.....	35
(4)	O'Reilly Discloses the Use of APIs	37
(5)	O'Reilly Discloses Using JavaScript to Create and Modify HTML Documents in Browsers, Including Interaction with Java—the Same Subject Matter Claimed in the '719 Patent	37
(6)	O'Reilly Describes JavaScript's Caching Functions, the Same Function Claimed As “Application-Independent” in the '719 Patent	39
(7)	O'Reilly Discloses “Event Handlers” that Provide Controller Logic.....	40
VIII. The Claims of the '719 Patent are Invalid.....		40
A.	Roy Anticipates All Elements of Independent Claims 1, 13, 25, and 26.....	40
B.	Roy Renders Obvious Independent Claims 1, 13, 25, and 26.....	55
C.	Roy, in Combination with Fennell and O'Reilly, Renders Obvious Dependent Claims 2, 4–12, 14, and 16–24	55
(1)	Motivation to Combine.....	56
(2)	Claims 2 and 14: “a HyperText Transport Protocol”	64
(3)	Claims 4 and 16: “partition/ing a screen area associated with the browser software into frames”	65
(4)	Claims 5 and 17: “wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are associated with at least one frame and one or more views for	

	display in accordance with the application are associated with at least another frame”	68
(5)	Claims 6 and 18: “wherein the at least one view frame is a visible frame”	72
(6)	Claims 7 and 19: “wherein the at least one frame associated with the at least a subset of the model, the view-generating logic and the controller logic is not a visible frame”	74
(7)	Claims 8 and 20: “form/ing at least one frame with which application-independent view-generating logic and controller logic is associated”	76
(8)	Claims 9 and 21: “wherein the at least one application-independent view-generating logic and controller logic frame further has an application-independent model associated therewith”	82
(9)	Claims 10 and 22: “wherein the at least one application-independent view-generating logic and controller logic frame serves as an application programming interface for developing views to be displayed in accordance with the application”	88
(10)	Claims 11 and 23: “wherein the views are implemented in accordance with the HyperText Markup Language and the application programming interface is implemented in accordance with the JavaScript language”	91
(11)	Claims 12 and 24: “wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand”	94
D.	Roy, in Combination with Fennell, Renders Obvious Claims 3 and 15 of the ’719 Patent.....	99
IX.	Secondary Considerations	100
X.	Conclusion	101

I, John K. Bennett, Ph.D., P.E., declare as follows:

I. Engagement

1. I have been retained by the law firm of Ballard Spahr LLP, who represents Carvana, LLC (“Carvana”) to provide expert testimony in support of Carvana’s Petition for *Inter Partes* Review of U.S. Patent No. 7,702,719 (“the ’719 Patent”), filed concurrently with this Declaration.

2. For my work related to this *Inter Partes* Review, I am being compensated at the rate of \$800 per hour. I have no financial interest in this proceeding, and my compensation is not conditioned on or impacted by the content of my testimony or the outcome of this proceeding.

II. Qualifications

3. I am a Professor Emeritus of Computer Science and Electrical and Computer Engineering at the University of Colorado at Boulder. Over my career, my research has focused broadly in the area of operating system/microprocessor architecture interaction, and more narrowly, in the area of distributed and parallel computing systems. My teaching has included graduate and undergraduate courses in computer architecture, operating system design, distributed systems, computer system design, and other topics.

4. My titles at the University of Colorado have included Director of Inworks, Associate Vice Chancellor for Innovation Initiatives, Professor of Computer Science at the University of Colorado Denver | Anschutz Medical Campus; and Archuleta Professor of Computer Science and Engineering and Professor of Electrical, Computer and Energy Engineering at the University of Colorado at Boulder. I served as the Director of the ATLAS Institute and as Associate Dean of Engineering for Education in the College of Engineering and Applied Sciences at the University of Colorado at Boulder.

5. I have a doctorate in Computer Science from the University of Washington. My doctoral dissertation involved the design, implementation and evaluation of a Smalltalk system distributed over a local area network. Smalltalk-80 is an object-oriented programming environment developed at the Xerox Palo Alto Research Center that was a precursor of Oracle's Java technology. I have done pioneering work in both distributed shared memory systems and adaptive software techniques for shared memory parallel computing. Prior to completing my thesis work at the University of Washington, I was President of Pacific Mountain Research, Inc. (PMR) in Seattle, Washington.

6. For the past forty-five years, I have focused on the development of computer software and hardware, both directly as a creator of sophisticated hardware and software systems, and as an academic, studying and contributing to the field. Examples of commercial and academic computer systems that I have helped develop include the mission control system for Jet Propulsion Laboratories, the control system for the Washington, D.C. Metropolitan Transit Authority light rail system, the propulsion control system for the Spruance class gas-turbine-powered destroyer, the Eden distributed computer system, an object-oriented distributed computing system developed at the University of Washington, several multiprocessor and virtual memory systems, several high performance graphics and image processing systems, several very large scale integrated (VLSI) circuits, biomedical instrumentation systems, and video processing systems. These systems were utilized or marketed by several companies, including Motorola, Signetics, RCA, IBM, Eastman Kodak, Microsoft, Sequent, Texas Instruments, Weyerhaeuser, the United States Department of Defense, and the New York Stock Exchange.

7. I have published over eighty research papers, and have given presentations at numerous conferences, universities, and research laboratories. A list of my publications can be found in Appendix A.

8. Over the years, I have received significant research funding from industry and government organizations in support of my work in over thirty-one research and educational projects, including support from the National Science Foundation, the Defense Advanced Research Project Agency, NASA, the National Institutes of Health, the National Institute of Disability and Rehabilitation Research, and the Texas Higher Education Coordinating Board, as well as grants from Compaq, Intel, Tandem, Microsoft, Ansell, Regent, Bell Northern Research, Hewlett-Packard, and IBM. A list of my grants can be found in Appendix A.

9. I have been an active member of the Association for Computing Machinery since 1979, and the Institute for Electrical and Electronic Engineers (in which I hold the rank of Life Senior Member) since 1976. I am the past president of the national Governing Board of Engineers Without Borders USA, and have received several teaching awards, including the W.M. Keck Foundation National Award for Engineering Teaching Excellence. I am a named inventor on two U.S. Patents.

10. As a result of this experience, I am thoroughly familiar with distributed and object-oriented system design and development, hardware and software development, and software testing and support. I have participated in the development and testing of commercial software products as both a software developer and manager. In the course of my research and teaching, I have also become broadly familiar with various client and server operating systems, and a variety of applications development and enterprise programming environments.

11. Over the past thirty plus years, I have been qualified as an expert in computer systems in courts through the United States and Canada, and have offered testimony on subjects relating to computer software in more than a dozen cases. My expert testimony has included analysis of computer software behavior, comparisons of functionally similar software and hardware, and patent claim interpretation. A list of the cases in which I have testified is contained in Appendix B.

III. Materials Reviewed

12. My knowledge and opinions in these areas derives in part from my active involvement in computer systems research and development throughout my career. As part of my teaching and academic advancement, I have kept current on issues in computer software, including operating and distributed system development.

13. Aside from my background in computer science, in the course of preparing this report I have taken into consideration the '719 Patent and its prosecution history, a paper authored by the inventors of the '719 Patent,¹ the exhibits listed in the Exhibit List filed with the petition for *inter partes* review of the '719 Patent, invalidity contentions related to the '719 Patent served in *Carvana, LLC v. International Business Machines Corp.*, Case No. 1:23-cv-8616-KMK-VR (S.D.N.Y.) ("the Related Litigation") as well as any other material referenced in this declaration.

IV. Legal Standards

14. I am not an attorney or legal expert. I do not and will not offer legal opinions. I have been instructed on matters of law relevant to forming my opinions in this matter. I have applied those instructions as described in this declaration.

¹ I understand that this paper, entitled *Developing Highly-Responsive User Interfaces with DHTML and Servlets*, Betz, et al., available at <https://dominoweb.draco.res.ibm.com/reports/rc21628.pdf> (Dec. 20, 1999) (Ex. 1014), was not disclosed by IBM to the USPTO during prosecution of the '719 Patent.

A. Claim Construction

15. I understand that claim construction is a matter of law. I have been informed by counsel of the legal standards that apply to claim construction, and I have applied them in forming my opinions.

16. I understand that in addition to the Related Litigation, IBM has asserted the '719 Patent against Zynga Inc., the gaming company.

17. I further understand that the parties in the Related Litigation agreed to certain constructions of terms/phrases in the '719 Patent, and I have applied those meanings when arriving at my opinions:

Claim Term(s) or Phrases(s)	Agreed Construction(s) for the Asserted Claims in the Related Litigation²	Relevant Claims in the '719 Patent³
"A method for use in a client/server system of reducing interactions between a client and server in association with an application being accessed by the client at the server . . ."	Preamble is limiting	1, 25, 26
"view generating . . . logic associated with the application"	program code that generates a screen or window representation of a subset of the model that the application chooses to display	1, 5, 12, 13, 17, 24–26
"the client and the server both locally . . . execute the view-generating and controller logic"	"view-generating logic" and "controller logic" executed on client must be the same "view-generating logic" and "controller logic" executed on	1, 13, 25, 26

² In the Related Litigation, I understand claims 1–4, 8, 10–12, and 26 were asserted ("the Asserted Claims"). I understand that all the '719 Patent claims, claims 1–26, are being challenged in this IPR.

³ The agreed constructions encompass only the Asserted Claims in the Related Litigation (claims 1–4, 8, 10–12, and 26).

Claim Term(s) or Phrases(s)	Agreed Construction(s) for the Asserted Claims in the Related Litigation²	Relevant Claims in the '719 Patent³
	the server, or some subset thereof	

18. I also understand that there are other terms in the '719 Patent on which the parties in the Related Litigation disagreed on the correct meaning. I have evaluated the '719 Patent claims, and under the meanings proposed by each party in the Related Litigation, my opinions do not change, as explained further below:

Claim Term(s) or Phrases(s)	Carvana's Proposed Constructions	IBM's Proposed Constructions Relevant to the Asserted Claims in the Related Litigation⁴	Relevant Claims in the '719 Patent
Application	"software that allows users to access and update data"	no construction necessary/plain and ordinary meaning	1, 3, 5, 8–13, 15, 17, 20–26
"model associated with [an/the] application"	"application data, rules, and algorithms affecting the application data"	"the data, rules, and algorithms affecting the data associated with the application" ⁵	1, 13, 25, 26
"controller logic associated with the application"	"program code that processes user requests and causes the model to be changed and the view to be refreshed"	"program code that processes user requests and causes the model to be changed and/or the view to be refreshed"	1, 13, 25, 26
frame(s)	"section(s) of a webpage, each constituting a distinct HTML document"	no construction necessary/plain and ordinary meaning	4–10, 16–22

⁴ IBM's constructions encompass only the Asserted Claims in the Related Litigation (claims 1–4, 8, 10–12, and 26).

⁵ I understand that this is the construction adopted by the Court in the Zynga matter involving accused gaming technology. *International Business Machines Corp. v. Zynga Inc.*, Case No. 22-590-GBW, D.I. 172 at 3 (D. Del. Oct. 30, 2023).

19. For any term or phrase not expressly mentioned above, I have given the term or phrase its plain and ordinary meaning as would be understood by a person of ordinary skill in the art at the time the '719 Patent was filed.

B. Anticipation

20. I understand that, following construction of the claims of a patent, determining anticipation of a patent claim requires a comparison of the properly construed claim language to the relevant prior art on a limitation-by-limitation basis.

21. I understand that a prior art reference “anticipates” an asserted claim, and therefore renders the claim unpatentable and invalid, if each and every element of the claim is disclosed in that prior art reference, either explicitly or inherently (i.e., necessarily present or necessarily implied).

22. I further understand that the petitioner needs to prove a proposition of unpatentability by a preponderance of the evidence.

C. Obviousness

23. I understand that a patent claim is obvious and therefore unpatentable and invalid if the subject matter of the claim, taken as a whole, would have been obvious to a person of ordinary skill in the art of the claimed subject matter as of the time of the invention. I understand that when assessing the obviousness of claimed subject matter, the following factors are evaluated: (1) the scope and content of the prior art; (2) the difference or differences between each claim of the patent and the prior art; (3) the level of ordinary skill in the art at the time the patent was filed; and (4) objective evidence of nonobviousness.

24. I understand that claimed subject matter may be obvious in view of one or more items of prior art. I understand, however, that it is not enough to show simply that all the limitations of the claimed subject matter are spread throughout various prior art. Rather, for claimed subject matter to be obvious in light of multiple references, there must be some reason(s) or motivation(s) for one of ordinary skill in the art to combine the prior art references to arrive at the claimed subject matter.

25. I understand that, in seeking to determine whether an invention that is a combination of known elements would have been obvious to a person of ordinary skill in the art at the time of the invention, the references must be considered in their entirety to determine whether the disclosures in such references render the combination obvious to a person of ordinary skill in the art at the time of the invention.

26. I understand that, while not required, the prior art references themselves may provide a teaching, motivation, or reason to combine. Other times the motivation to link two or more prior art references may stem from common sense to a person of ordinary skill in the art at the time of the invention.

27. I understand that a particular combination may be considered obvious by showing that it was obvious to try the combination. I understand that, if a technique has been used to improve one device, and if a person of ordinary skill in the art would recognize that it could improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill.

28. I understand that the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results.

29. I understand that a patent is likely to be invalid for obviousness if a person of ordinary skill can implement a predictable version or if there existed at the time of the invention a known problem for which there was an obvious solution encompassed by the patent's claims. Further, I understand that a patent or publication is considered analogous art even if it is in a different field of endeavor, where it is reasonably pertinent to the particular problem with which the inventor was involved or which the invention attempts to solve. I understand that when a work is available in one field of endeavor, design incentives or commercial inducements may prompt variations of that work, either in the same field or a different one.

30. I understand that combining embodiments related to each other in a single prior art reference would not ordinarily require a leap of inventiveness.

31. I understand that one of ordinary skill in the art must have had a reasonable expectation of success when combining references for claimed subject matter to be obvious.

32. I understand that one cannot use hindsight to determine that an invention was obvious.

33. Finally, I understand that the petitioner needs to prove a proposition of unpatentability by a preponderance of the evidence.

D. Level of Skill in the Art

34. I understand that the level of ordinary skill in the relevant art at the time of the invention is relevant to inquiries such as the meaning of claim terms, the meaning of disclosures found in the prior art, and the reasons one of ordinary skill in the art may have for combining different references.

35. I understand that factors that may be considered in determining the level of ordinary skill include: (1) the education of the inventor; (2) the type of problems encountered in the art;

(3) prior art solutions to those problems; (4) rapidity with which innovations are made; (5) sophistication of the technology; and (6) education level of active workers in the relevant field.

I have been further informed and understand that a person of ordinary skill in the art is also a person of ordinary creativity.

36. In my opinion, a person of ordinary skill in the art in the technology related to the '719 Patent generally would have had at least a bachelor's degree in computer science, computer engineering, or a related field, with at least two years of professional experience in the fields of networking and network-based systems or applications, such as client-server and web-based systems, or an equivalent level of skill, knowledge, and experience. Additional relevant work experience would substitute for the educational background, and additional relevant educational experience would substitute for the work experience.

V. Background of the Technology

37. The following sections of my Declaration will provide background information on the technology relevant to the '719 Patent, specifically the Model-View-Controller (MVC) paradigm and JavaScript.

A. Model-View-Controller (MVC) and "Dual-MVC" Were Well Known Years Before the '719 Patent Was Filed

38. Model-View-Controller (MVC) is a nearly forty-year-old object-oriented user interface paradigm that divides the user interface into three related components: Model, View, and Controller.

39. MVC was not novel at the time the '719 Patent was filed. Indeed it was described in, and an essential element of, an object-oriented programming language that I worked on in the 1980s called Smalltalk-80.⁶

40. MVC consists of three object classes: the *Model*, which maintains the “state” of an application, the *View*, which creates a representation of that state for viewing, and the *Controller*, which manages user interaction between the Model and the View (i.e., updates to the application’s state).

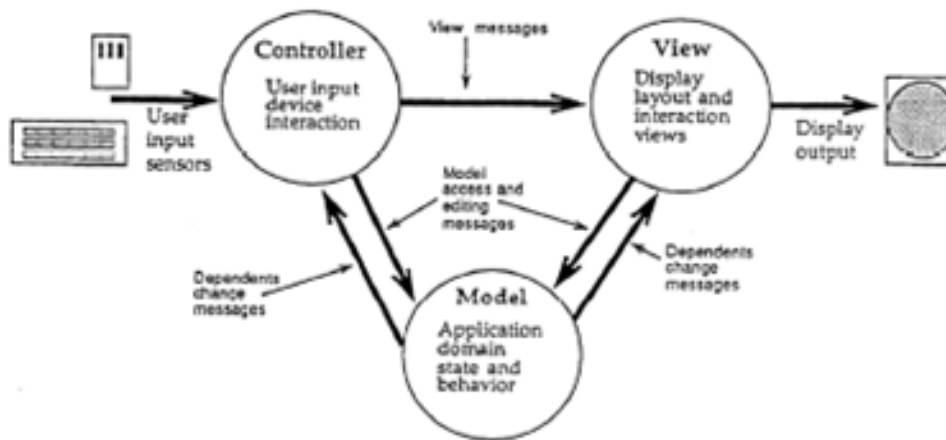
41. Controllers communicate with the Model, and provide the interface between the Model, its associated Views (of which there may be several), and the user interface.

42. As a simple example of MVC, a Model is a collection of data items (e.g., a list of library books); a View creates representations of such data (e.g., sorting the books by author, title, or subject); and a Controller provides mechanisms for manipulating the model (e.g., checking out a book, or adding a book to the library). Controllers typically maintain the interface between their associated Models and Views and the user interface devices (keyboard, pointing device, etc.). Views and Controllers generally have exactly one Model, but a Model can be associated with several Views and Controllers.

43. Applications in Smalltalk-80 typically had a single Model, and several view/controller pairs. Views requested data from their model, and created representations of objects’ data that they received for subsequent display. These relationships are shown in the figure below⁷:

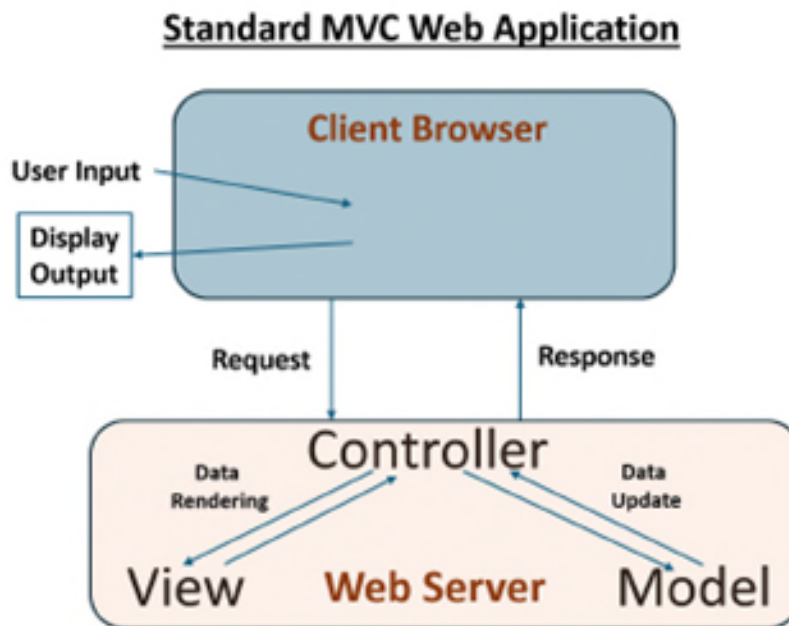
⁶ G. Krasner & S. Pope, A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System, *Journal of Object-Oriented Programming*, Vol. 1, p.5, Jan. 1988 (Ex. 1008).

⁷ G. Krasner & S. Pope, A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System, *Journal of Object-Oriented Programming*, Vol. 1, p.5, Jan. 1988 (Ex. 1008).



44. As shown in the figure above, Smalltalk-80 display output was managed by a View.

45. MVC's simplicity, encapsulation, and modularity made it a popular choice for developers designing early web applications with the client browser serving as the user interface, depicted here:



46. As developers created mechanisms for embedding code within MVC-based browsers (e.g., JavaScript) in the 1990s, developers divided MVC responsibilities between server and client. The concept of doing so is referred to in the '719 Patent as “dual MVC.”

47. “Dual-MVC” concepts were not new at the time of the ’719 Patent, and were disclosed in, for example, U.S. Patent 5,838,909 (Roy),⁸ which I discuss more in depth below at §VII.A. Further, any distributed implementation of Smalltalk-80 necessarily used “dual-MVC,” because an interactive application running on a distributed implementation of Smalltalk-80 would have to have model, view, and controller components on every machine running the application^{9,10,11}. Having model, view, and controller elements on the client and server was not inventive and a POSITA would have considered the concept routine at the time the ’719 Patent was filed.

48. Although “dual-MVC” distributed applications had been built prior to MVC use in web application development (see examples above), “dual-MVC” became more widespread with the advent of web application software environments that facilitated such development.

B. JavaScript Was Known at Least as of 1995

49. As HTML evolved during the 1990s, developers looked for ways to create dynamic components on the client after a web page was loaded into the browser. This desire to add dynamic components to static HTML text led to the development of what came to be known as JavaScript.

50. JavaScript was first released in 1995 as part of Netscape Navigator. JavaScript let web developers create webpages with client-side interactivity, and when used with HTML,

⁸ U.S. Patent No. 5,838,909 (“Roy”) (Ex. 1004).

⁹ John K. Bennett, The Design and Implementation of Distributed Smalltalk, In *Proceedings of the Second ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 318-330, Orlando, Florida, October 1987. Special Issue of SIGPLAN Notices, Volume 22, Number 12, December, 1987. (Ex. 1009).

¹⁰ D. Decouchant, Design of a Distributed Object Manager for the Smalltalk-80 System. In *Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 444-452, Portland, Oregon, October 1986. Special Issue of SIGPLAN Notices, Volume 21, Number 11, November, 1986. (Ex. 1010).

¹¹ P. L. McCullough, Transparent Forwarding: First Steps, In *Proceedings of the Second ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 331-341, Orlando, Florida, October 1987. Special Issue of SIGPLAN Notices, Volume 22, Number 12, December, 1987. (Ex. 1011).

JavaScript enabled developers to create dynamic webpages. Using JavaScript, developers could create dynamic browser programs with fewer server interactions than traditional CGI methods.¹²

51. By the time Netscape released Navigator 3.0 in 1996, JavaScript could communicate with the Java interpreter built into the browser and could work with and control Java applets running within a web page. Although JavaScript and Java were two distinct programming languages created by different organizations, these capabilities led to Java and JavaScript being referred to as a “good team.”¹³

VI. The ’719 Patent Claims the Well-Known “Dual-MVC” Paradigm

A. Overview

52. In the ’719 Patent, the applicants claim placing elements of the model, view, and controller on the client and server improves application performance by reducing the number of server interactions:

The invention relates to web applications and, more particularly, to an architecture and method of programming which improves performance and eases the development and maintenance effort. The invention addresses performance by employing a dual-MVC approach, in which a subset of the application’s Model-View-Controller reside on the client, and the full Model-View-Controller and View-Generating Logic reside on the server, **thereby reducing the number of required server interactions.**¹⁴

53. While portions of the application’s model, view, and controller reside on the client, the server always contained the authoritative version of the application’s model, view and controller—a concept a POSITA readily knew by the time of the ’719 Patent’s filing.¹⁵ The idea

¹² *JavaScript: The Definitive Guide* (Flanagan, David) (Willison, Frank, Ed.) (O’Reilly & Associates, Inc., pub.) (2d. Ed. Jan. 1997) (ISBN: 1-56592-234-4) (“O’Reilly”) (Ex.1006), 20 (§1.3.3). Page cites correspond to Exhibit 1006’s pagination.

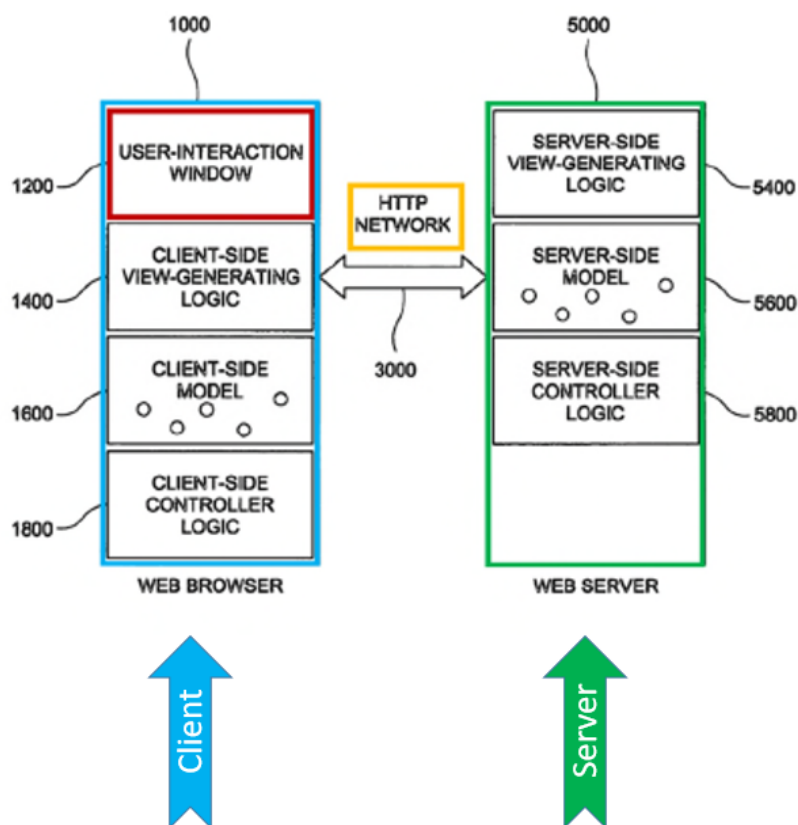
¹³ O’Reilly (Ex.1006), 17–18 (§1.2.1).

¹⁴ ’719 Patent (Ex. 1001) at 2:39–49 (emphasis added).

¹⁵ ’719 Patent (Ex. 1001) at 2:42–47.

of placing model, view, and controller elements on both the client and the server (which the '719 Patent calls “dual-MVC”) was not novel, but was disclosed in art predating the '719 Patent, such as Roy described herein.

54. Figure 3 of the '719 Patent, depicts “dual-MVC” as follows:¹⁶



55. In this figure, the “User Interaction Window” 1200 runs within a web browser, but all other boxes are the individual components of the Model-View-Controller found on both client (surrounded by a blue box) and server (surrounded by a green box). The “HTTP Network” 3000 is presumed to be a standard network interface capable of supporting HTTP. The server-side MVC components (5000) are considered to be the “authoritative” (complete) copies of these

¹⁶ '719 Patent (Ex. 1001) at Fig. 3 (coloration added).

components; the client-side versions (1000) are subsets of the server version (“a subset of the application’s Model-View-Controller reside on the client, and the full Model-View-Controller and View-Generating Logic reside on the server”).¹⁷

56. The ’719 Patent claims the use of JavaScript, yet acknowledges that “JavaScript can be used to implement much of the same functionality that [prior art] applets provide[.]”¹⁸

57. The ’719 Patent states that when using “dual-MVC,” “the client machine maintains part of the Model, eliminating many round-trips to the server”¹⁹ because the client has stored some model data locally—a well-known concept.

58. The ’719 Patent further claims HTML “frames”/“framesets” to implement “dual MVC,” i.e., storing a portion of the model, view-generating logic, and controller logic on both the server and client. But HTML frames/framesets were also well known at the time the ’719 Patent was filed, as references such as O’Reilly (1997) describe, to be discussed herein.

59. An HTML “frameset” is a collection of frames arranged on a web page in some sort of grid, each of which could contain additional framesets in a hierarchical manner. When using frames, before the ’719 Patent was filed, developers commonly embedded JavaScript code on what were called “invisible frames” (frames with locations outside the browser window or dimensions small enough or to make them not visible on a screen). This code was used to update the visible frames. By putting this code in an **invisible** frame, a programmer avoided the problem of overwriting application code anytime the visible frame was modified or reloaded.

¹⁷ ’719 Patent (Ex. 1001) at 2:43–46.

¹⁸ ’719 Patent (Ex. 1001) at 3:1–13; 2:22–23.

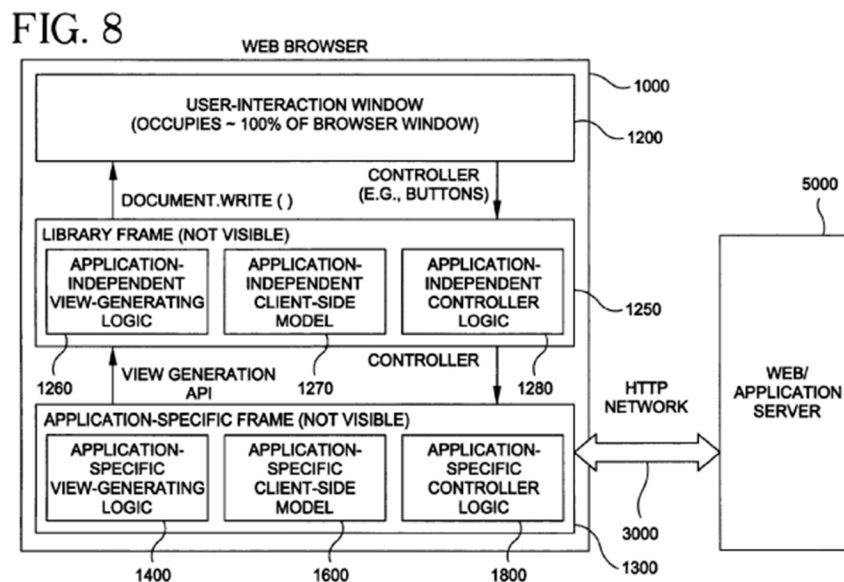
¹⁹ ’719 Patent (Ex. 1001) at 3:14–22.

60. The use of invisible frames was well known at the time of filing of the '719 Patent.

As described below in §VII.C, O'Reilly (1997) a developer could use HTML to create invisible frames to associate with logic that could dynamically update visible frames:

As we saw in the error handler example, JavaScript code in one window (or frame) can dynamically create an HTML document in another window (or frame). . . . But what if your design calls for all the frames to be dynamic? A static frame is still required, but the trick here is to create the static frame so that it is invisible! You do this by explicitly creating it at a location that is greater than 100% of the frame width or height.²⁰

61. As in O'Reilly (1997), the '719 Patent describes a “not visible” “library frame” that provides an API implementation for using JavaScript to generate an HTML document.²¹



62. As the '719 Patent describes, this non-visible library frame contained “application-independent” MVC components:

²⁰ O'Reilly (Ex. 1006) at § 12.4 (highlighting added).

²¹ '719 Patent (Ex. 1001) at Fig. 8 and 7:51–58.

“In a preferred embodiment, the library frame contains the application-independent view-generating logic and application-independent controller logic and may contain an application-independent model.”²²

63. The '719 Patent provides examples of its claimed “application-independent client-side model” as “caching and data-movement operations,” and “application independent controller logic” as “generic user-interface event handling.”²³²⁴ But caching and data-movement operations were well known and practiced long before the '719 Patent was filed, and even described in a 1971 patent issued to IBM.²⁵

64. The '719 Patent’s example of “application independent controller logic” is “generic user-interface event handling,” and “application independent view-generating logic” is “logic to generate a button on the screen.” But so too were user-interface event handling, and placing user interface components within a web browser screen well known and practiced long before the '719 Patent was filed.

B. '719 Patent Claims

65. The '719 Patent contains 4 independent claims: Claims 1, 13, 25, and 26. The independent claims overlap in many respects.²⁶ The '719 Patent claims the well-known concepts of having a model, view-generating logic, and controller logic on the server and client, via well-known means to do so, like JavaScript, HTML, and invisible frames.

	Claim 1	Claim 13	Claim 25	Claim 26
Preamble	A method for use in a client/server system of	A network-based system, comprising	An article of manufacture for use in a client/server	A method for use in a client/server system of reducing interactions between

²² '719 Patent (Ex. 1001) at 8:41–44.

²³ '719 Patent (Ex. 1001) at Fig. 8 and 7:51–58.

²⁴ '719 Patent (Ex. 1001) at 8:44–49.

²⁵ U.S. Patent No. 3,771,137 (“Barner”) (Ex. 1013).

²⁶ '719 Patent (Ex. 1001) at claims 1, 13, 25, and 26.

	Claim 1	Claim 13	Claim 25	Claim 26
	reducing interactions between a client and server in association with an application being accessed by the client at the server, the method comprising the steps of		system for reducing interactions between a client and server in association with an application being accessed by the client at the server, comprising machine readable media containing one or more programs which when executed implement the steps of:	a client and server in association with an application being accessed by the client at the server, the method comprising the steps of
(a)	configuring the server to store a model associated with the application and to execute view- generating and controller logic associated with the application; and	a server having at least one processor operative to: (i) store a model associated with an application associated with the server, and (ii) execute view-generating and controller logic associated with the application; and	configuring the server to store a model associated with the application and to execute view-generating and controller logic associated with the application; and	configuring the server to: (i) store a model associated with the application; (ii) execute view-generating logic associated with the application; and (iii) execute controller logic associated with the application; and
(b)	configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic	a client, coupled to the server via a network, having at least one processor operative to: (i) store at least a subset of the model associated with the application; and	configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic	configuring the client to: (i) store at least a subset of the model associated with the application; (ii) execute at least a subset of the view-generating logic associated with the application; and (iii) execute at least a

	Claim 1	Claim 13	Claim 25	Claim 26
	associated with the application;	(ii) execute at least a subset of the view-generating and controller logic associated with the application;	associated with the application;	subset of the controller logic associated with the application;
(c)	wherein one or more portions of the application are performed at the client without the client having to interact with the server, and	wherein one or more portions of the application are performed at the client without the client having to interact with the server such that interactions between the client and server are reduced, and	wherein one or more portions of the application are performed at the client without the client having to interact with the server, and	wherein one or more portions of the application are performed at the client without the client having to interact with the server, and
(d)	further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.	further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.	further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.	further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith;
(e)				further wherein, in accordance with such a dual model-view-controller arrangement, a model comprises application data, rules, and algorithms affecting the data, a view comprises a screen or window representation of a

	Claim 1	Claim 13	Claim 25	Claim 26
				subset of the model that the application chooses to display, and a controller comprises the logic that processes user requests, and causes the model to be changed and the view to be refreshed.

66. The dependent claims of the '719 Patent include various limitations on the general “dual-MVC” model embodied in the independent claims. For example, claims 2 and 14, which depend from claims 1 and 13 respectively, require that client and server (claim 2) and the network (claim 14) communicate over/are “a HyperText Transport Protocol network.”²⁷

67. Claims 3 and 15, which similarly depend from claims 1 and 13 respectively, require that the client/client processor perform the “one or more portions of the application in accordance with browser software running thereon.”²⁸

68. Claim 4, which depends from claim 3, requires that the “configuring step further comprises the step of partitioning a screen area associated with the browser software into frames,” though the claim does not explain to which of the two “configuring steps” ((a) or (b)) this refers.²⁹

69. Claim 16 requires that the “client processor is further operative to partition a screen area associated with the browser software into frames.”³⁰

²⁷ '719 Patent (Ex. 1001) at claims 2 and 14.

²⁸ '719 Patent (Ex. 1001) at claims 3 and 15.

²⁹ '719 Patent (Ex. 1001) at claim 4.

³⁰ '719 Patent (Ex. 1001) at claim 16.

70. Claims 5 and 17, which depend from claims 4 and 16, respectively, require that “wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are associated with at least one frame and one or more views for display in accordance with the application are associated with at least another frame.”³¹

71. Claims 6 and 18, which depend from claims 5 and 17, respectively, require that “wherein the at least one view frame is a visible frame.”³²

72. Claims 7 and 19, which depend from claims 5 and 17, respectively, require that “wherein the at least one frame associated with the at least a subset of the model, the view-generating logic and the controller logic is not a visible frame.”³³

73. Claim 8, which depends from claim 4, requires “wherein the configuring step further comprises forming at least one frame with which application-independent view-generating logic and controller logic is associated,” though the claim does not explain to which of the two “configuring steps” ((a) or (b)) this refers.³⁴

74. Claim 20, which depends from claim 16, requires “wherein the client processor is further operative to form at least one frame with which application-independent view-generating logic and controller logic is associated.”³⁵

75. Claims 9 and 21, which depend from claims 8 and 20, respectively, require “wherein the at least one application-independent view-generating logic and controller logic frame further has an application-independent model associated therewith.”³⁶

³¹ ’719 Patent (Ex. 1001) at claims 5 and 17.

³² ’719 Patent (Ex. 1001) at claims 6 and 18.

³³ ’719 Patent (Ex. 1001) at claims 7 and 19.

³⁴ ’719 Patent (Ex. 1001) at claim 8.

³⁵ ’719 Patent (Ex. 1001) at claim 20.

³⁶ ’719 Patent (Ex. 1001) at claims 9 and 21.

76. Claims 10 and 22, which depend from claims 8 and 20, respectively, require “wherein the at least one application-independent view-generating logic and controller logic frame serves as an application programming interface for developing views to be displayed in accordance with the application.”³⁷

77. Claims 11 and 23, which depend from claims 10 and 22, respectively, require “wherein the views are implemented in accordance with the HyperText Markup Language and the application programming interface is implemented in accordance with the JavaScript language.”³⁸

78. Claims 12 and 24, which depend from claims 1 and 13, respectively, require “wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.”³⁹

VII. Specific Examples of Prior Art Disclosures

A. U.S. Patent No. 5,838,909 (Roy)

(1) Roy Is Analogous Prior Art to the '719 Patent

79. U.S. Patent 5,838,909 (Roy) discloses an Internet-based application that limits server round trips by having a model, view-generating logic, and controller logic on both client and server, or “dual-MVC” application.⁴⁰ Roy is entitled “Reducing Latency When Synchronizing Access to a Multi-User Database Over a Network.” Roy was filed May 23, 1996, and issued Nov. 17, 1998.⁴¹ Because its filing date precedes that of the '719 Patent, I understand that Roy meets

³⁷ '719 Patent (Ex. 1001) at claims 10 and 22.

³⁸ '719 Patent (Ex. 1001) at claims 11 and 23.

³⁹ '719 Patent (Ex. 1001) at claims 12 and 24.

⁴⁰ U.S. Patent No. 5,838,909 (“Roy”) (Ex. 1004).

⁴¹ Note: In some instances, Roy uses the terms “Master” and “Slave” to refer to components of the described invention. These terms, which trivialize the horrific practice of treating people as property, are not part of the modern engineering

the legal standards required to qualify as prior art to the '719 Patent. Roy is neither identified on the face of the '719 Patent, nor was it considered during prosecution.⁴²

80. The '719 Patent's stated problem was a need for Internet-based "applications which [do] not require continual network communication between the client and server."⁴³ A POSITA seeking to reduce client/server interactions would look to Roy because Roy provides a way to reduce latency when synchronizing access to a networked database.

81. Roy's system reduces client/server interactions by enabling both server and client to process events locally whenever possible. Roy discloses a distributed software architecture where a model is held on a "master" computer system (in what was called the "master database"), and copies of that model are stored on "slave" computer systems (in what was called the "slave database").

(2) Roy Discloses a "Dual-MVC" Networked System

82. Roy discloses Internet-connected computers with stored application software, for example, a simulated American football computer game.⁴⁴ In the game, the player controlling the offense controls more factors like speed and direction, whereas a player controlling the defense may only need to select a predefined defensive pattern.⁴⁵ In that case, the player on offense would have a more restrictive latency requirement.⁴⁶ Roy recognizes that the computer employed by the user with the most restrictive latency requirement should be the "master event server" (i.e., the

lexicon. Since these terms are essential to being able to describe the patent in the language in which it was written, I will, where necessary, also use them here.

⁴² '719 Patent File History (Ex. 1002).

⁴³ '719 Patent at 1:48–2:35.

⁴⁴ Roy, Fig. 1, 3:23–35; 3:48–56 (Ex.1004).

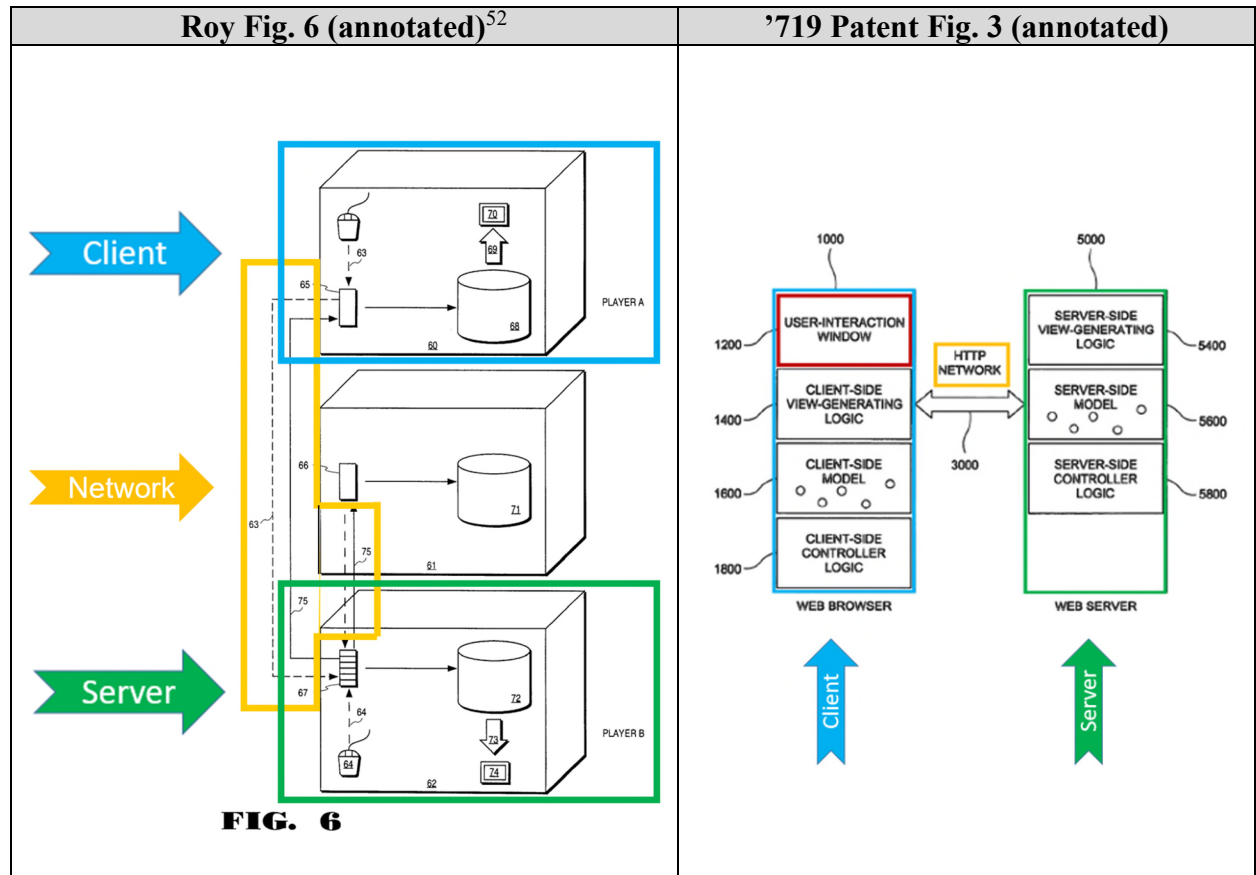
⁴⁵ Roy, 3:23–35 (Ex.1004).

⁴⁶ Roy, 3:23–35 (Ex.1004).

server) in the system.⁴⁷ Every other player's computer is the "slave event server," (i.e., a client).⁴⁸ Any user's computer could be the "server" or the "client,"⁴⁹ as in the '719 Patent.⁵⁰

83. Roy discloses a system where certain events were processed locally, thus reducing the number of server interactions.

84. Roy shows a model, view, and controller on a server and a client.⁵¹



⁴⁷ Roy at 4:37–43 (Ex.1004).

⁴⁸ Roy at 4:37–43 (Ex.1004).

⁴⁹ Roy describes that the master event server is migrated to the machine used by the user with the least restrictive latency requirement, as depicted in Figure 9A. Ex.1004, 5:65–6:2. The process begins when the current master event server receives a request in the form of a slave migration event (step 901), and if certain criteria are met in steps 902 and 903, the master event server (or "server," in '719 Patent parlance) changes to a slave event server (or "client," in '719 Patent parlance). Ex.1004, Fig.9A, 7:60–8:5.

⁵⁰ '719 Patent (Ex. 1001), Fig.2; 4:33–36.

⁵¹ Roy, 5:37–49 (Ex.1004).

⁵² Components described in green correspond to the server; those in blue correspond to the client. The yellow polygon surrounds the network connection.

85. Like the “dual-MVC” **client** and **server** in ’719 Patent Figure 3, the preferred embodiment of Roy shown in Roy Figure 6 shows a **client** and **server**. In Roy, the computer system containing the master event server and master database are considered the “server.”⁵³ Any non-server computer system (i.e., those with a slave database and slave event server) are “clients.” Further, like the ’719 Patent’s description of a model, view-generating logic, and controller logic existing on both the **client** and **server**, Roy also describes these same three components: (1) the **model** in client-side **slave copy database 68** and server-side **database 72**; (2) the **view-generating logic** in client-side **rendering commands 69** and **display 70** and server-side **rendering commands 73** and **display 74**; and (3) the **controller logic** in client-side generated **events 63** and **event server 65** and server-side generated **events 64** and **event server 67**.^{54,55}

86. Player A in Figure 6 demonstrates the client side of Roy’s “dual-MVC” architecture. Player A can locally play the game with minimal server interaction because Player A’s **rendering commands 69** generate **display 70** of the game (the “view” of the “dual-MVC”) from the **slave copy 68** of the master database (the “model” of the “dual-MVC”), rather than the master database directly.⁵⁶ Updating the local view from local rendering commands reduces bandwidth consumption on the network.⁵⁷

⁵³ Roy (Ex.1004), Figures 7-10.

⁵⁴ Roy (Ex. 1004), Fig.6, 8:60–9:5.

⁵⁵ While one embodiment of Roy described in Fig.5A uses computer 61 “initially” as a “central server,” in Fig.6 the master event server and master database (i.e., the “server”) have migrated to the most active user, and **computer 62** is now the server. Ex.1001, Fig.5A, 5:17–36; 8:60–9:5. For purposes of comparing the claim elements of the ’719 Patent to Figure 6, **computer 62** is the “server,” and **computer 60** is the “client.” The initial hosting of the master event server and the master database on computer 61 allows the game system of Roy to be operational prior to users joining the game. After users join the game, Roy teaches the master database and master event server (i.e., the “server”) migrating to the computer system of the most active user.

⁵⁶ Roy (Ex. 1004), 5:29–35. While this description is in reference to Fig.5A, 5:17–36, it also applies to Fig.6.

⁵⁷ Roy (Ex. 1004), 5:29–33.

87. The client controller logic (“control” of the “dual-MVC”) handles Player A’s input events 63 (such as mouse button clicks). Slave event server 65 receives Player A’s input and then based on a determination from the local controller logic, either retains and handles the events locally, or passes them along to the master event server 67.⁵⁸

88. Handling events locally minimizes server interaction. Locally retained events are those “purely local” “events which set display options or other functions that are only relevant to” Player A’s computer.⁵⁹ Non-local events, i.e., events received by the master event server, are used to update the master database.⁶⁰ This architecture reduces server interaction—the same goal the ’719 Patent describes and claims—because not all events travel to the master server, only those deemed to be master events.

89. Player B in Figure 6 demonstrates the server side of Roy’s “dual-MVC” architecture. While playing, Player B’s rendering commands 73 generate display 74 of the game (the “view” of the “dual-MVC”) from master database 72 (the “model” of the “dual-MVC”).⁶¹

90. The server controller logic (“control” of the “dual-MVC”) handles Player’s B input events 64 (such as mouse button clicks). Master event server 67 receives Player B’s input and then passes the events to slave event servers.⁶²

91. The ’719 Patent describes an online “help desk.”⁶³ There, a “consultant” assisting a “customer” opens an interface and asks for information like ID, operating system, and

⁵⁸ Roy (Ex. 1004), 5:43–46; 8:60–9:5 (noting that in Fig.6, the master event server is 67).

⁵⁹ Roy (Ex. 1004), 5:40–43.

⁶⁰ Roy (Ex. 1004), 5:37–49.

⁶¹ In Figure 6, database 72 is the master database. Roy (Ex. 1004), 8:52–9:5 (referencing Figure 6 after Player B hosts the master event server); 4:44–55 (noting there is only one master database in the system that is preferably on the same system as the master event server); 5:29–33 (referring to the Figure 5A embodiment).

⁶² Roy (Ex. 1004), 5:46–49.

⁶³ ’719 Patent (Ex.1001), 6:1–45.

hostname.⁶⁴ In the prior art embodiment, each request required a round trip to the server to populate corresponding data.⁶⁵ The '719 Patent's claimed improvement is storing more corresponding data on the consultant computer (client) to eliminate server roundtrips.⁶⁶ The '719 Patent explains that when "a subset of the Model is stored on the client," "a number of interactions with the server can be eliminated, thus providing better performance."⁶⁷ But this is exactly what Roy discloses. In Roy, each client (e.g., Player A in Figure 6) has a copy of the Model ([slave copy of the master database 68](#)), thereby eliminating server roundtrips.⁶⁸

92. Thus, Roy discloses a client/server system each with a **model**, **view-generating logic**, and **controller logic**, consistent with the '719 Patent's "dual-MVC" disclosure.

B. U.S. Patent No. 5,695,400 (Fennell)

(1) Fennell Is Analogous Prior Art to the '719 Patent

93. The concept of playing multiplayer games over the Internet using a browser was not unique to Roy, but instead, for example, U.S. Patent No. 5,695,400 (Fennell) describes this well-known concept.⁶⁹ Fennell is entitled "Method of Managing Multi-Player Game Playing Over a Network." Fennell was filed January 30, 1996, and issued December 9, 1997. Because its filing date in 1996 precedes that of the '719 Patent in 2000, I understand that Fennell meets the legal standards required to qualify as prior art to the '719 Patent. Fennell is neither identified on the face of the '719 Patent, nor was it considered during prosecution.

⁶⁴ '719 Patent (Ex.1001), 6:4–31.

⁶⁵ '719 Patent (Ex.1001), 6:33–35.

⁶⁶ '719 Patent (Ex.1001), 6:35–45.

⁶⁷ '719 Patent (Ex.1001), 6:35–45.

⁶⁸ Roy (Ex. 1004), Fig.6.

⁶⁹ U.S. Patent No. 5,695,400 (Fennell) (Ex. 1005).

94. Fennell is in the same field of endeavor as the '719 Patent, as both employ what the '719 Patent called "dual-MVC." Both disclose an Internet-based system that reduces host computer interactions, contemplates differences in connection speed between the client and server and "compensates for network latency among different terminals."⁷⁰ As Fennell recognizes, network latency effects could make a game "less enjoyable and therefore less popular," and thus Fennell's invention sought to minimize the effects of variation of network latency.⁷¹

95. In Fennell's traditional game play, the method comprises (1) transmitting a representation of a game challenge to the terminals via the network; (2) receiving a game response signal from one or more computers; (3) assigning each responding terminal a rank in a priority scheme in accordance with a characteristic of their respective game response signals; (4) determining which responding terminal had the highest rank; and (5) sending signals to the responding terminals assigning them respective degrees of control of the game in accordance with their respective ranks.⁷²

96. While traditional gameplay requires interaction with a host computer, Fennell discloses a "dual-MVC" embodiment wherein host computer interaction is minimized via equipping each user with a model (game challenge), a view (display) and controller (terminal equipped with an input/output device):

[T]erminals 110 may be programmed so that one of the terminals 110 transmits the game challenges. Such embodiments would permit players to manage their own games without the need for active management by the host computer 114. For example, in a trivia game, at the beginning of the game the host computer 114 may download an encrypted question-and-answer file to the terminal 110 of a particular player (e.g., a player who initiates the game by issuing an electronic-mail or "chat" challenge to other players). This player at this terminal may issue a game challenge

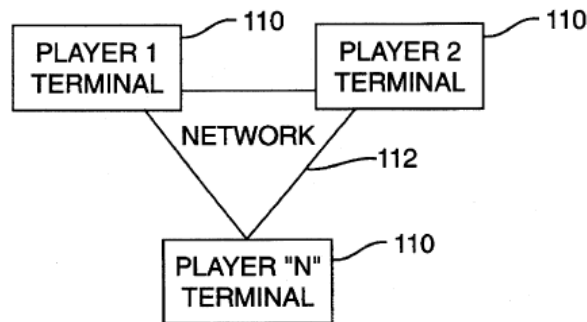
⁷⁰ '719 Patent at 3:6-10; Fennell (Ex. 1005) at Abstract, 1:39-43; 3:1-2.

⁷¹ Fennell (Ex. 1005) at 1:25-36.

⁷² Fennell (Ex. 1005) at 2:39-56.

by pressing an appropriate function key on his or her terminal 110, whereupon that player's terminal 110 both transmits the game challenge over the network 112 to the other terminals and displays the game challenge.⁷³

97. In that embodiment, host computer interaction is minimized as terminal 110 downloads an encrypted question-and-answer file (the model), and terminal 110 sends that model to additional participants. The additional participants do not interact with the host computer, but only terminal 110 via the network 112. Each additional participant therefore would have a copy of the model, a display (view) and controller (terminal equipped with “any suitable input/output device”⁷⁴):



75

98. Accordingly, like the '719 Patent and Roy, Fennell discloses a distributed system with each terminal having a model, view, and controller.

(2) Fennell Is Related to Roy and Discloses Distributed Internet Applications Accessed Via Browser Software

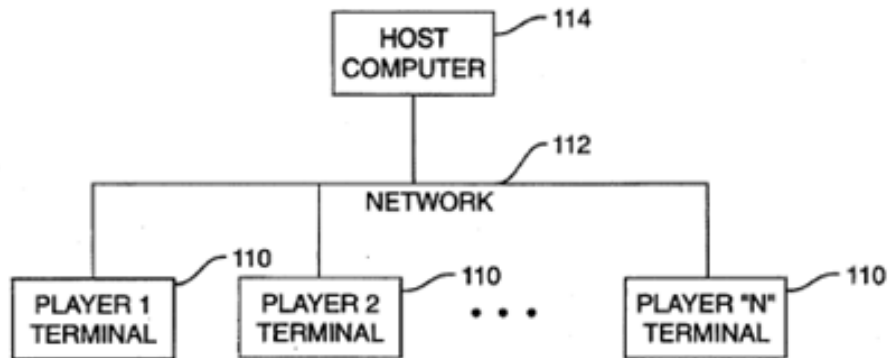
99. Figure 1 of Fennell describes an illustrative method of the invention, including a host computer and a plurality of terminals connected by the Internet:⁷⁶

⁷³ Fennell (Ex. 1005) at 5:38–53.

⁷⁴ Fennell (Ex. 1005) at 3:3–4.

⁷⁵ Fennell (Ex. 1005) at Fig. 3.

⁷⁶ Fennell (Ex. 1005) at Fig. 3; Fennell (Ex. 1005) at 3:23–30 (the “network” included “the Internet”).



100. The Fennell “terminal” refers to any suitable input/output device such as “an appropriately programmed computer with a keyboard and display; a personal digital assistant (PDA) with a touchscreen; a telephone with a keypad and a receiver; a teletype; a television with a remote control (the well-known ‘set-top box’ concept) and the like.”⁷⁷ Fennell also teaches that the terminal could be “a dedicated device with programming permanently ‘burned into’ a read-only memory (ROM), **a smart terminal such as a World Wide Web browser with software ‘applets’ (written in, e.g., Sun Microsystems’ Java language) downloaded on demand**, or a general-purpose computing device with suitable game programming that is conventionally loaded from a hard disk, CD-ROM, network server, or other mass storage device when the game is to be played.”⁷⁸ Indeed, Fennell suggests that many embodiments would take the form of “a dedicated device,” or “a smart terminal such as a World Wide Web browser with software ‘applets’” downloaded on demand because of the low marginal cost of distributing software, which would inform a POSITA seeking to distribute the system Roy discloses.⁷⁹

⁷⁷ Fennell (Ex. 1005) at 3:3-8.

⁷⁸ Fennell (Ex. 1005) at 3:11-19 (emphasis added).

⁷⁹ Fennell (Ex. 1005) at 3:19-22.

101. A POSITA would appreciate that Fennell's disclosure of a browser-based application over the Internet necessarily discloses the use of HTML in the browser. Further, as I have already described, at the time of the '719 Patent, Java applets were well known and could be controlled by JavaScript.

102. Accordingly, Fennell discloses a browser-enabled online application that minimized client/server interaction and, as noted above, disclosed an embodiment that, like Roy's system, allowed users to serve as the "host," minimizing host computer/server interaction.

C. JavaScript: The Definitive Guide ("O'Reilly")

(1) O'Reilly, a Printed Publication and Analogous Prior Art to the '719 Patent, Discloses the Use of Frames, Invisible Frames, HTML, JavaScript, and APIs Years before the '719 Patent Was Filed

103. Claims 4 and 16 of the '719 Patent, and each of the claims dependent thereon, claim the use of frames in connection with an MVC approach. However, the use of frames, invisible frames, HTML, JavaScript, and API controls to execute client-side programs was well known in the art and described in a book published in January 1997, entitled JAVASCRIPT: THE DEFINITIVE GUIDE (Flanagan, David) (Willison, Frank, Ed.) (O'Reilly & Associates, Inc., pub.) (2d. Ed. Jan. 1997) (ISBN: 1-56592-234-4) ("O'Reilly").⁸⁰ Because the publication date of O'Reilly in 1997 precedes the filing date of the '719 Patent in 2000, I understand that O'Reilly meets the legal standards required to qualify as prior art to the '719 Patent. O'Reilly is neither identified on the face of the '719 Patent, nor was it considered during prosecution.

104. O'Reilly comprehensively describes JavaScript and its application to web-based applications. This included JavaScript running in a browser, JavaScript running on a server,

⁸⁰ O'Reilly (Ex. 1006).

JavaScript and Java running together and interacting in a browser, JavaScript and Java running together and interacting on a server, as well as various combinations of these configurations.⁸¹

105. O'Reilly was one of a series of well-known and inexpensive books that described current (at the time) technology and best practices across a wide range of topics in computer systems and programming. These books were routinely distributed as “swag” at relevant conferences and technical meetings, and frequently purchased by computer scientists and engineers interested in best practices.

106. I still own and refer to many of the O'Reilly series of books. A POSITA at the time of the '719 Patent would have likely read and consulted O'Reilly routinely.

107. I also have been informed that the United States Patent and Trademark Office's Precedential Opinion Panel issued an opinion finding another book in the O'Reilly series—SED AND AWK (Tim O'Reilly, ed., O'Reilly & Associates, Inc.) (Nov. 1990)—to be a printed publication, noting that the book is a textbook from an established publisher, O'Reilly, and a well-known book series. *JavaScript: The Definitive Guide* is part of this well-known book series.

108. The '719 Patent cites and incorporates by reference another book in the O'Reilly series, *Dynamic HTML, The Definitive Reference*.⁸²

109. The '719 Patent applicants did not disclose O'Reilly during prosecution of the '719 Patent, despite the inventors being aware of the series.⁸³

⁸¹ O'Reilly (Ex. 1006).

⁸² '719 Patent at 2:15–20 (Ex.1001).

⁸³ See *Developing Highly-Responsive User Interfaces with DHTML and Servlets*, Betz, et al., available at <https://dominoweb.draco.res.ibm.com/reports/rc21628.pdf> (Dec. 20, 1999) (Ex. 1014) (citing JAVASCRIPT: THE DEFINITIVE GUIDE (Flanagan, David) (O'Reilly, pub.) (3d. Ed. 1998); '719 Patent (Ex. 1001) at 2:15–20 (citing O'Reilly series).

110. The '719 Patent is directed to using HTTP⁸⁴ (e.g., Claims 2 and 14), frames (e.g., Claims 4–6, 8–10, 16–18, 20–22), invisible frames (e.g., Claims 7 and 19), HTML and JavaScript (e.g., Claims 11 and 23), API controls (e.g., Claims 10 and 22), and application independent features like caching, view-generating logic, and controller logic (e.g., Claims 8–10, 20–22). O'Reilly discloses the use of HTTP, frames, invisible frames, HTML and JavaScript, API controls, and features like caching, view-generating logic, and controller logic, making it analogous art.

(2) O'Reilly Demonstrates a POSITA's Familiarity with HTTP as of at Least 1997

111. Though communication over HTTP is claimed in claims 2 and 14 of the '719 Patent, communication over HTTP was not novel at the time the '719 Patent was filed. O'Reilly demonstrates that by the time it was published in 1997, readers were well familiar with HTTP networks. For example, O'Reilly teaches using cookies to “provide the state information that is missing from the stateless HTTP protocol of the Web.”⁸⁵ When discussing implementations of Server-Side JavaScript, O'Reilly further explains how to handle certain HTTP requests⁸⁶ and the implementation of cookies as “an extension to the HTTP protocol.”⁸⁷ In short, O'Reilly demonstrates that HTTP, as claimed in claims 2 and 14, was well known by at least 1997.

⁸⁴ I understand that the Examiner during prosecution of the '719 Patent took “Official Notice” that “HTTP is well-known in the art as taught by Gish [U.S. Patent No. 5,768,510, filed on July 1, 1996], wherein Gish teaches the client and server communicate over a HyperText Transport Protocol network[.]”

⁸⁵ O'Reilly (Ex. 1006) at 21 (§1.3.5).

⁸⁶ O'Reilly (Ex. 1006) at 32 (§1.6.5).

⁸⁷ O'Reilly (Ex. 1006) at 234 (§15.1).

(3) O'Reilly Discloses Using Frames and Invisible Frames to Generate HTML Documents and Store View-Generating and Controller Logic, the Same Subject Matter Claimed to Be (But Was Not) Invented in the '719 Patent

112. Claims 4–10 and 16–22 of the '719 Patent purport to claim frames and invisible frames, as well as view-generating logic and controller logic. But O'Reilly describes the well-known concept of using JavaScript to create views of and control multiple frames in HTML documents years before the '719 Patent was filed.

113. O'Reilly explains that it was then known that Navigator (and other browsers) could “display multiple pages within a single window by using *frames*—a feature that allows a single browser window to be divided horizontally or vertically (or both) into individual sections that each display a separate HTML document.”⁸⁸ O'Reilly further explains that the same concepts that were used to create multiple *windows* of HTML documents also applied to creating multiple *frame* windows, and in fact, at that time, it was “much more common to create a web site that use[d] multiple frames than it [was] to create one that use[d] multiple browser windows.”⁸⁹

114. O'Reilly also teaches the use of **invisible** frames to dynamically update HTML documents; subject matter claimed to be (but was not) invented in the '719 Patent.

115. For example, after teaching that a programmer could use JavaScript to create dynamic HTML documents in separate windows or frames, O'Reilly recognizes the problem of using JavaScript to dynamically create an HTML document its *own* window or frame, “because this generally overwrites the JavaScript code itself!”⁹⁰

⁸⁸ O'Reilly (Ex. 1006) at 184 (§11.3).

⁸⁹ O'Reilly (Ex. 1006) at 207 (§12.4).

⁹⁰ O'Reilly (Ex. 1006) at 208 (§12.4); '719 Patent at 3:47–50 (“a client-side application attempting to update its View will destroy itself by overwriting its own Controller logic and Model data”).

116. To avoid overwriting, O'Reilly recommends the use of invisible frames:

If your web page design calls for one static frame and two frames that have their contents dynamically updated, the static frame can contain the JavaScript code necessary to update the dynamic frames. But what if your design calls for all the frames to be dynamic? **A static frame is still required, but the trick here is to create the static frame so that it is invisible!** You do this by creating it at a location that is greater than 100% of the frame width or height.⁹¹

117. O'Reilly even provided sample code to deploy the technique of creating an invisible frame:⁹²

Example 12.5: Creating an Invisible Frame

```
<!-- Create two frames that take up half the screen each, and one that
-->
<!-- takes up "all the rest" of the room. The third frame will be -->
<!-- invisible, because it has a height of zero. -->
<frameset rows="50%,50%,*">
  <!-- first two frames start out empty, loading no documents -->
  <frame name="dynamic_frame_1">
  <frame name="dynamic frame 2">
  <!-- invisible frame contains the code that will -->
  <!-- dynamically update the others -->
  <frame name="invisible_frame" src="program.html">
</frameset>
```

118. O'Reilly notes that JavaScript can be used to generate documents, and enables users to write arbitrary HTML into a document as the document was being parsed by the browser. That meant that a programmer could use JavaScript to specify view-related features of an HTML document, like background and text color. O'Reilly further explains that this technique “works particularly well in multiframe documents.”⁹³

⁹¹ O'Reilly (Ex. 1006) at 208–09 (§12.4) (emphasis added).

⁹² O'Reilly (Ex. 1006) at 208–09 (§12.4; Example 12.5) (emphasis added).

⁹³ O'Reilly (Ex. 1006) at 19 (§1.3.1).

119. In fact, O'Reilly teaches that a programmer would have to use both JavaScript and HTML when creating framed layouts.⁹⁴ O'Reilly explains that JavaScript provides control over which web pages are displayed in particular windows or frames of the browser.⁹⁵

(4) O'Reilly Discloses the Use of APIs

120. Though the '719 Patent claimed the use of JavaScript APIs as novel, O'Reilly disclosed such use years earlier. For example, O'Reilly discloses that Java can interact with JavaScript "through the API provided by the JScript class."⁹⁶ Further, it was (and still is) good software engineering practice to create API libraries when developing application software.

(5) O'Reilly Discloses Using JavaScript to Create and Modify HTML Documents in Browsers, Including Interaction with Java—the Same Subject Matter Claimed in the '719 Patent

121. The '719 Patent claims the use of HTML and JavaScript together in claims 11 and 23. But O'Reilly describes the use of JavaScript in web browsers, the relationships between JavaScript, HTML, and Java, and exemplary "typical" uses of JavaScript years before the '719 Patent was filed.⁹⁷

122. O'Reilly explains that JavaScript is an object-oriented programming language that allows for executable content to be included on web pages, such that a web page did not need to be static but could include dynamic programs that interact with a user, control the browser, and dynamically create HTML content.⁹⁸ O'Reilly explains that JavaScript can be used to interact with

⁹⁴ "JavaScript does not define methods that allow you to directly create and manipulate frames within a browser window. However, the ability to dynamically generate HTML allows you to programmatically write the HTML tags that will create any desired frame layout." O'Reilly (Ex. 1006) at 19 (§1.3.1).

⁹⁵ O'Reilly (Ex. 1006) at 19 (§1.3.1).

⁹⁶ O'Reilly (Ex. 1006) at 307 (§19.6.3).

⁹⁷ O'Reilly (Ex. 1006) at 4–9 (Table of Contents).

⁹⁸ O'Reilly (Ex. 1006) at 14 (§1).

HTML form documents and control the behavior of those forms by defining “event handlers” for the HTML forms that respond to, for example, button clicks.⁹⁹ O’Reilly explains that the “interactions between JavaScript code and the web browser and the browser’s contents are what matter most” and that mastering the ability for JavaScript to interact with content on a web page and communicate with Java applets and plug-ins was the real key to using JavaScript effectively.¹⁰⁰

123. Despite the ’719 Patent’s attempts to distinguish its invention from the capabilities of Java applets, the ’719 Patent acknowledges that “JavaScript can be used to implement much of the same functionality that applets provide.”¹⁰¹ O’Reilly also describes mechanisms for Java applets and JavaScript to interact. O’Reilly explains that JavaScript communicates with the Java interpreter built into the browser and can work with and control Java applets in a web page. While O’Reilly notes that Java and JavaScript are separate programming languages, it explains that both can deliver “executable content” over networks and that JavaScript and Java “make a good team.”¹⁰²

124. O’Reilly also explains that JavaScript in some ways was more practical than traditional CGI scripts, because “Client-side JavaScript can also perform preprocessing of input data, which can reduce the amount of data that must be transmitted to the server.”¹⁰³ Another “powerful” piece of JavaScript technology is its ability to “read and write properties of, and invoke methods of, Java applets and plug-ins. This capability is tremendously powerful, and truly allows JavaScript to ‘script’ Java.”¹⁰⁴ While O’Reilly explains the differences between Client-Side

⁹⁹ O’Reilly (Ex. 1006) at 15 (§1.1).

¹⁰⁰ O’Reilly (Ex. 1006) at 17 (§1.1).

¹⁰¹ ’719 Patent at 2:22–23.

¹⁰² O’Reilly (Ex. 1006) at 17–18 (§1.2.1).

¹⁰³ O’Reilly (Ex. 1006) at 20 (§1.3.3).

¹⁰⁴ O’Reilly (Ex. 1006) at 21 (§1.3.6).

JavaScript and Server-Side JavaScript, it notes that an “obvious capability of server-side JavaScript is to dynamically generate HTML to be displayed by the client.”¹⁰⁵

(6) O’Reilly Describes JavaScript’s Caching Functions, the Same Function Claimed As “Application-Independent” in the ’719 Patent

125. The ’719 Patent discloses that caching is an example of an application-independent model. But years earlier, O’Reilly disclosed that allowing JavaScript files to be cached by the browser allows them to load much more quickly.¹⁰⁶ O’Reilly further describes the use of storing “cookies” so that JavaScript can recall certain properties.¹⁰⁷

126. O’Reilly also teaches a method of caching images for easy replacement into dynamic HTML pages:

The ability to dynamically replace one image in a static HTML document with another image opens the door to any number of special effects, from animation, to images that change when clicked on, to “digital clocks” that update themselves in real time. With a bit of thought, you can probably imagine many more potential uses for this technique. In order to make the image replacement technique viable, and in order to make animations and other special effects responsive enough to be useful, we need some way to ensure that the necessary images are loaded into the browser’s cache.¹⁰⁸

127. Thus, the use of caching as an application-independent model was well known by the time the ’719 Patent was filed.

¹⁰⁵ O’Reilly (Ex. 1006) at 32 (§1.6.5).

¹⁰⁶ O’Reilly (Ex. 1006) at 164–66 (§10.1.2, 10.2).

¹⁰⁷ O’Reilly (Ex. 1006) at 234–41 (§15.1–15.5).

¹⁰⁸ O’Reilly (Ex. 1006) at 242 (§16.1).

(7) O'Reilly Discloses “Event Handlers” that Provide Controller Logic

128. The '719 Patent claims “controller logic,” but O'Reilly discloses controller logic in the form of event handlers years before the '719 Patent was filed.

129. More specifically, O'Reilly explains that JavaScript could be used to create event handlers (controller logic) to be used in client-side JavaScript. Event handlers could be associated with a button in an HTML form, for example, which could then trigger a piece of JavaScript code.¹⁰⁹ O'Reilly characterizes event handlers as “almost always the central element of any interesting JavaScript program.”¹¹⁰

130. In short, O'Reilly discloses the use of HTTP, frames, invisible frames, HTML and JavaScript, API controls, caching, view-generating logic, and controller logic— the same subject matter claimed in the limitations of dependent claims 2–12 and 14–24, as further explained below.

VIII. The Claims of the '719 Patent are Invalid

131. In my opinion, every claim of the '719 Patent is invalid, on the grounds enumerated below.

A. Roy Anticipates All Elements of Independent Claims 1, 13, 25, and 26

132. In my opinion, Roy anticipates each limitation of Claims 1, 13, 25, and 26.

	Claim 1	Claim 13	Claim 25	Claim 26
Preamble	A method for use in a client/server system of reducing interactions between a client and server in association with an	A network-based system, comprising	An article of manufacture for use in a client/server system for reducing interactions between a client and server in association with an	A method for use in a client/server system of reducing interactions between a client and server in association with an

¹⁰⁹ O'Reilly (Ex. 1006) at 108; 166–71 (§6.5; §10.3).

¹¹⁰ O'Reilly (Ex. 1006) at 252 (§17.1).

	Claim 1	Claim 13	Claim 25	Claim 26
	application being accessed by the client at the server, the method comprising the steps of		application being accessed by the client at the server, comprising machine readable media containing one or more programs which when executed implement the steps of:	application being accessed by the client at the server, the method comprising the steps of

133. In my opinion, Roy discloses “a method for use” (claims 1 and 26) and “an article of manufacture” (claim 25) in a “client/server system” (claims 1, 25 and 26) or “network system” (claim 13) (e.g., the client/server and network systems depicted in Figure 6) of “reducing interactions between a client and server” (claims 1, 25 and 26) (e.g., Roy discloses that “local events” are processed locally) “in association with an application being accessed by the client at the server” (e.g., a user connects to an online application).

134. First, Roy discloses “a client/server system,” which is also “a network-based system”: “FIG. 6 illustrates a multi-player game running in a network in which a slave event server has established a direct connection to the master event server.”¹¹¹ “FIG. 4B illustrates a multi-player game running in a network using a client-server configuration.”¹¹²

135. Second, Roy discloses a method for “reducing interactions between a client and server in association with an application being accessed by the client” (relevant to claims 1, 25, and 26). Roy discloses multiple examples of this method. For one, Roy explains how providing

¹¹¹ Roy at 2:38–40 (Ex. 1004).

¹¹² Roy at 2:31–32 (Ex. 1004).

slave copies of the database to every non-master computer connected to the system reduces the need to send rendering commands across the network:¹¹³

“By providing slave copies of the database in every other machine involved in the game, the need to send rendering commands across the network is reduced or eliminated, which reduces bandwidth consumption.”¹¹⁴

136. “Local events” are processed on the client and therefore do not require any server interaction.¹¹⁵ Accordingly, just as claimed in the preamble (claims 1, 25, and 26), interactions between the client and server are reduced.

137. Two, Roy discloses that an “event” is created each time a user issues a command and if the event is determined to be a local event, then it is processed by the client and the results displayed, thereby eliminating the need for server interaction.¹¹⁶ Examples of local events include “events which set display options or other functions that are only relevant to [that] computer system.”¹¹⁷

138. Three, in Roy’s Figure 6 embodiment, Player B’s computer is the master server (“server” in ’719 Patent parlance) and *all* events generated by Player B are processed locally:¹¹⁸

¹¹³ Roy at Fig. 5A and 5:17–49 (Ex. 1004).

¹¹⁴ Roy at 4:48–54 and Fig 5A (Ex. 1004).

¹¹⁵ Roy at Fig. 5A and 5:17–49 (Ex. 1004).

¹¹⁶ Roy at Fig. 5A (Ex. 1004).

¹¹⁷ Roy at 5:39–43 (Ex. 1004).

¹¹⁸ Roy at 6:5–14 (Ex. 1004); Figure 6 (annotated).

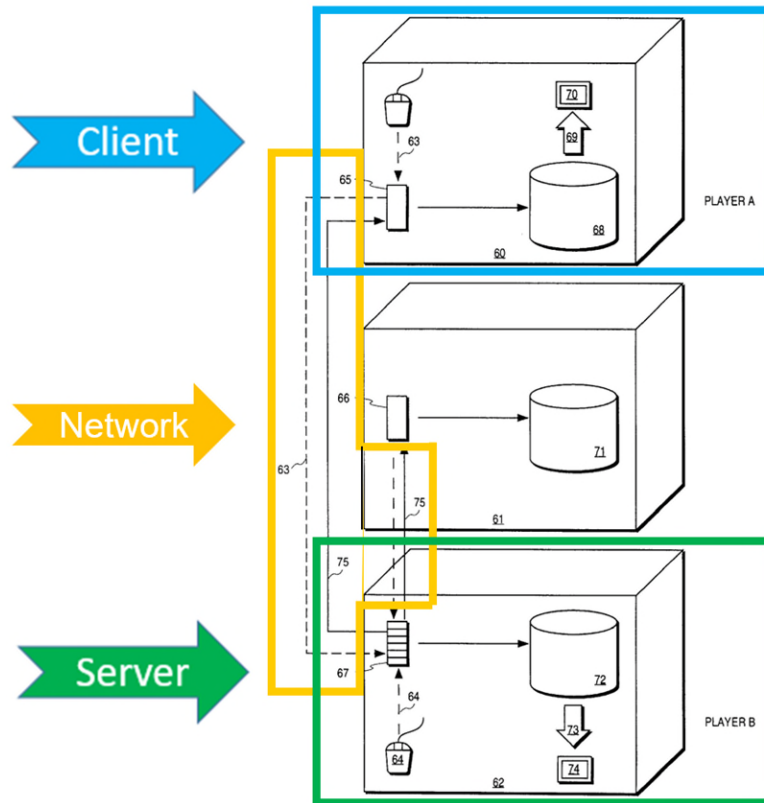


FIG. 6

139. Third, Roy discloses an “application being accessed by the client” (e.g., [computer system 60](#)) at “the server” (e.g., [master event server 67](#) on [computer 62](#)). In other words, “FIG. 6 illustrates a multi-player game running in a network in which a [slave event server](#) has established a direct connection to the [master event server](#).”¹¹⁹

140. This is accomplished by “machine-readable media containing one or more programs,” which, when executed, implement the application.

141. Accordingly, Roy discloses each limitation of the preamble of claims 1, 13, 25 and 26. I discuss limitation (a) next.

	Claim 1	Claim 13	Claim 25	Claim 26
(a)	configuring the server to store a	a server having at least one processor	configuring the server to store a	configuring the server to: (i) store a model

¹¹⁹ Roy at 2:38–41 (Ex. 1004).

	Claim 1	Claim 13	Claim 25	Claim 26
	model associated with the application and to execute view-generating and controller logic associated with the application; and	operative to: (i) store a model associated with an application associated with the server, and (ii) execute view-generating and controller logic associated with the application; and	model associated with the application and to execute view-generating and controller logic associated with the application; and	associated with the application; (ii) execute view-generating logic associated with the application; and (iii) execute controller logic associated with the application; and

142. In my opinion, Roy discloses “configuring the server” (e.g., the components in [computer 62](#)) “to store a model associated with the application” (e.g., [database 72](#) with the world model for the application) “and to execute view-generating and controller logic associated with the application,” as those terms have been construed and proposed for construction. This would necessarily require a system in which the server has at least one processor capable of conducting these actions. Thus, in my opinion (and as explained below), Roy discloses:

- configuring the server (e.g., [computer 62 with the master event server and master database](#)) to store the data, rules, and algorithms affecting the data associated with the application (e.g., [database 72](#) with the world model for the application) and application data, rules, and algorithms affecting the application data (e.g., [database 72](#) with the world model for the application);¹²⁰
- configuring the server (e.g., [computer 62 with the master event server and master database](#)) to execute program code that processes user requests (e.g., [event server 67](#) processing [events 64](#)) and causes the model to be changed **and/or** the view to be refreshed (e.g., [database 72](#) with the world model for the application is changed and the [display 74](#) is updated by [rendering commands 73](#));¹²¹ and

¹²⁰ This disclosure encompasses the constructions proposed in the Related Litigation.

¹²¹ My opinions do not change if the “and/or” (as proposed by IBM in the Related Litigation) or “and” (as proposed by Carvana in the Related Litigation) applies.

- configuring the server (e.g., **computer 62 with the master event server and master database**) to execute program code that generates a screen or window representation of a subset of the model that the application chooses to display (e.g., the server generates an updated **display 74** via **rendering commands 73**).¹²²

143. First, Roy discloses limitation (a) “configuring the server to store a model associated with the application.” For example, Roy configures the “master database” to store a “database represent[ing] a world model for the game” (the server).¹²³ Roy further explains how the “game world can be modeled by a database” which can be located on a user’s computer (the client).¹²⁴ Thus, whichever construction is adopted for “the model associated with the application,” the “configuring the server” limitation is met because the database contains the model of the application that is required to operate the application software.

144. Second, Roy discloses configuring **the server** to “execute view-generating . . . logic associated with the application.”

145. For example, in Figure 6, **computer 62** is the server and contains the master event server and master database.¹²⁵ It receives **events 64** from player B on **computer 62** (and from other users), updates **master database 72**, and generates **rendering commands 73** that update the **display 74**.¹²⁶ This functionality, described in relation to Figure 5A, is illustrated in Figure 6:

¹²² I understand this encompasses the construction agreed to by the parties in the Related Litigation.

¹²³ Roy at Abstract, 4:8-40 (Ex. 1004).

¹²⁴ Roy at 1:28-30 (Ex. 1004).

¹²⁵ Roy at 6:1-11 (Ex. 1004).

¹²⁶ Roy at Figure 6 (annotated) and 4:50-55, 5:27-36, 7:1-8.

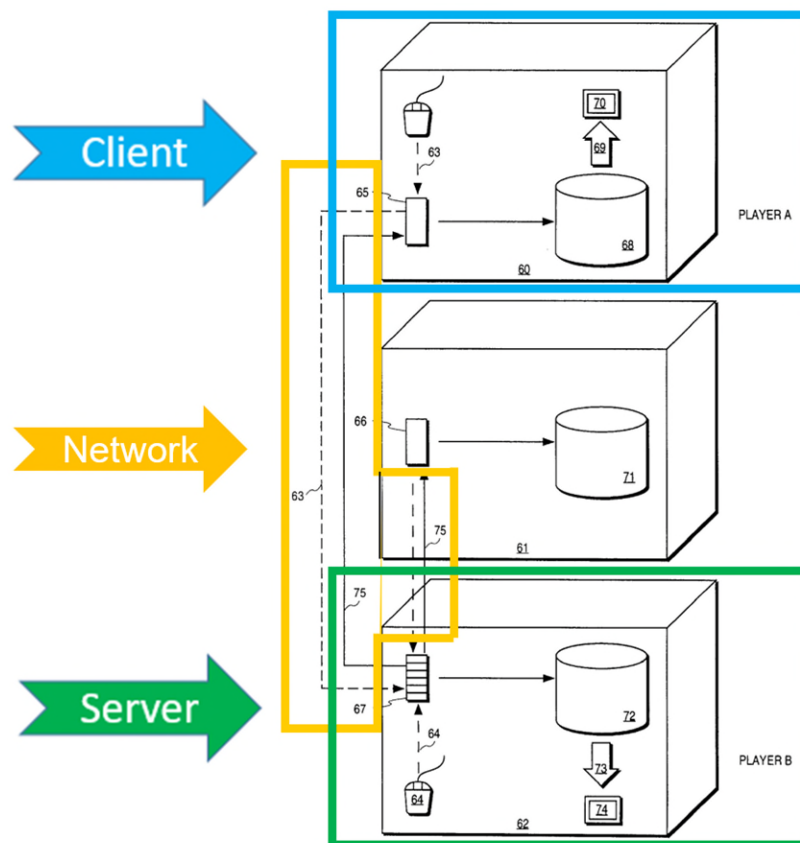


FIG. 6

146. Server Computer 62 executes view-generating logic when it creates rendering commands 73 used to update the display 74 for Player B.

147. Third, Roy discloses the server “execut[ing] . . . controller logic associated with the application.” For example, Roy explains that the “master event server sequences user inputs, or events, for updating the master database” (i.e., controller logic is executed to update the application model):

“A method and apparatus for reducing network latency during execution of a multiple-player game across a computer network are provided. A master database represents a world model for the game, and a master event server sequences user inputs, or events, for updating the master database. The master event server and the master database are initially located on a particular node in the network. Every other node that is used by a player in the game has a slave event server and a slave database.”¹²⁷

¹²⁷ Roy; Abstract (highlighting added) (Ex.1004).

148. In Figure 6, server **computer 62** receives **events 64** from Player B. Roy discloses how the user can issue commands and thereby generate “events” using a mouse, joystick, trackball, keyboard, or any other I/O device.¹²⁸ **Computer 62’s event server 67** then executes controller logic to update **master database 72** and causes **rendering commands 73** to be rendered and shown to the user on the user’s **display 74**.¹²⁹ Because **computer 62** is the server in the client/server system, Roy discloses configuring the server to store and execute controller logic.

149. Accordingly, Roy discloses every limitation (a) of the independent claims 1, 13, 25, and 26. I discuss limitation (b) of these claims next.

	Claim 1	Claim 13	Claim 25	Claim 26
(b)	configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application;	a client, coupled to the server via a network, having at least one processor operative to: (i) store at least a subset of the model associated with the application; and (ii) execute at least a Subset of the view-generating and controller logic associated with the application;	configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application;	configuring the client to: (i) store at least a subset of the model associated with the application; (ii) execute at least a subset of the view-generating logic associated with the application; and (iii) execute at least a subset of the controller logic associated with the application;

150. In my opinion, Roy discloses “configuring the client” (e.g., **computer system 60**) “to store at least a subset of the model associated with the application” (e.g., **slave copy of the master database 68**) and “to execute at least a subset of the view-generating and controller logic

¹²⁸ Roy at 3:66–4:13 (Ex.1004).

¹²⁹ Roy at Figure 6 and 5:27-49 (Ex 1004).

associated with the application,” under either party’s proposed construction in the Related Litigation. Thus, in my opinion (and as explained below), Roy discloses:

- configuring the client (e.g., [computer system 60](#)) to store at least a subset of the data, rules, and algorithms affecting the data associated with the application (e.g., [slave copy of master database 68](#)) and application data, rules, and algorithms affecting the application data (e.g., [slave copy of the master database 68](#));¹³⁰
- configuring the client (e.g., [computer system 60](#)) to execute program code that processes user requests (e.g., [event server 65](#) processing [events 63](#)) and causes the model to be changed and/or the view to be refreshed (e.g., [events 63](#) cause [slave copy of the master database 68](#) to be changed and [display 70](#) is refreshed to reflect the change);¹³¹
- configuring the client (e.g., [computer system 60](#)) to execute program code that generates a screen or window representation of a subset of the model that the application chooses to display (e.g., client [computer system 60](#)’s [rendering commands 69](#) generate an updated [display 70](#)); and
- that the view-generating logic and controller logic executed on the client is the same view-generating logic and controller logic executed on the server, or some subset thereof (e.g., the same view-generating logic and controller logic would be executed on each computer system connected to the network).

151. First, Roy discloses “configuring the client to store at least a subset of the model associated with the application.” In particular, Roy discloses that “[s]lave copies of the master database (or portions thereof) are stored” on each client computer.¹³² As I explained previously, the databases of Roy represent the world model for the application. Because the clients contain a copy of the master database (or a portion thereof), the client is configured to store at least a subset of the model.

¹³⁰ This disclosure encompasses the constructions proposed in the Related Litigation.

¹³¹ My opinions do not change if the “and/or” (as proposed by IBM in the Related Litigation) or “and” (as proposed by Carvana in the Related Litigation) applies.

¹³² Roy at 4:44-48 (Ex. 1004).

152. Thus, as an example, the entire master database “or portions thereof”¹³³ would be saved in database 68 of client computer 60, meaning at least a subset of the data comprising the world model would be stored on the client. This limitation is therefore met whether a “model” is “application data, rules, and algorithms affecting the application data” or “the data, rules, and algorithms affecting the data associated with the application,” because the database contains the model of the application that is required to interact with the application.

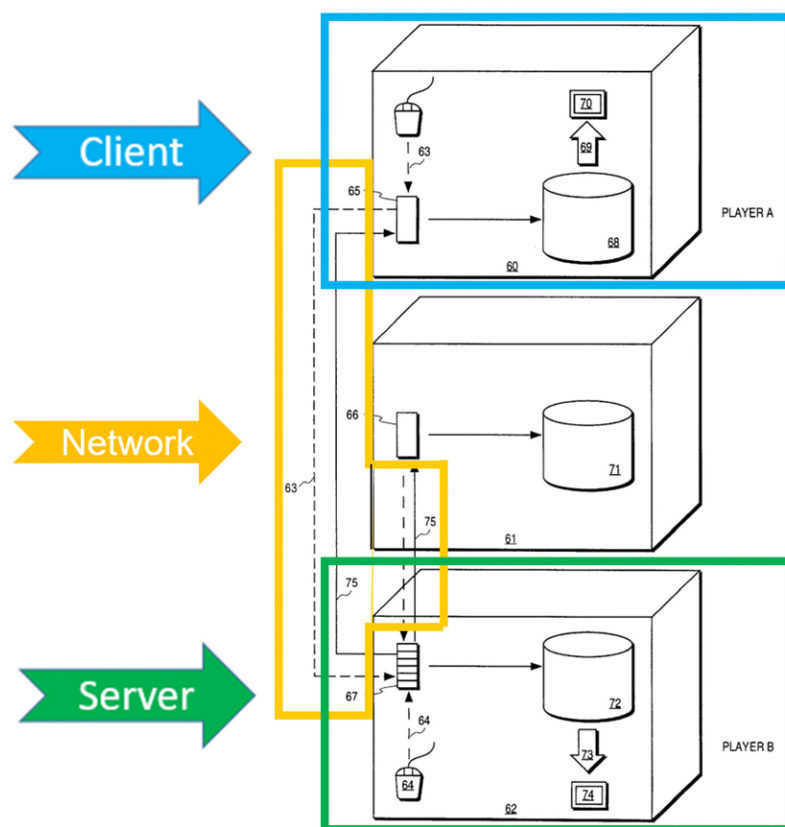


FIG. 6

134

153. Second, Roy discloses “configuring the client to execute at least a subset of the view-generating . . . logic associated with the application.” As an example, Roy discloses that the display 70 of client computer 60 is updated by rendering commands 69 from the slave copy 68 of

¹³³ Roy at 4:47-48 (Ex. 1004).

¹³⁴ Roy at Fig.6 (annotated) (Ex.1004).

the master database, rather than from the master database itself, thereby reducing server interactions on the network. While this description references Figure 5A, the elements relevant to computer 60 (player A) are consistent with respect to Figure 6.

154. Further, a POSITA would understand that the view-generating logic and controller logic would inherently be the same on each computer for efficiency and economic reasons. It would be particularly inherent to a POSITA given that Roy specifically addresses online applications wherein each user sees the same world. It would not make sense to use different logic to generate views of the same world for each computer. This is especially true given that Roy teaches that client and server computers have the same architecture: “FIG. 2 illustrates an example of an architecture which may be used for any of the client computer systems 2 or the server computer system 1.”¹³⁵

155. A POSITA would understand that Roy’s “view-generating logic” and “controller logic” are the same whether on the server or client, as any POSITA would, whenever technically feasible, seek to avoid the expense and inefficiency of developing different logic and program code to perform the same task, as a matter of both common sense and sound software engineering practice. This kind of “code reuse” has been routinely taught to computer science undergraduate students for at least as long as I have been teaching.

156. Third, Roy discloses “configuring the client to execute at least a subset of the . . . controller logic associated with the application” on the client. Roy discloses that Player A’s client computer system 60 executes at least a subset of the controller logic. Client computer 60 processes events 63 generated by user commands and event server 65 determines whether the events should be passed along to the master server, or whether they are “local events” used to update the local

¹³⁵ Roy at Figure 2 and 3:57-59 (Ex. 1004).

model stored in database 68 and generate rendering commands 69 causing display 70 to be refreshed. As Roy explains:

Located in computer system 60 (used by player A) are a slave event queue 65 and a slave copy 68 of master database 71. . . . [T]he display 70 associated with computer system 60 is updated by rendering commands 69 from slave copy 68 of master database 71, rather than directly from master database 71. . . . Purely local events might be, for example, events which set display options or other functions that are only relevant to computer system 62. Similarly, events 63 originating from computer system 60 are received first by slave event server 65 and then passed along to master event server 66, unless those events are purely local to computer system 60.¹³⁶

157. While this description references Figure 5A, the elements relevant to computer 60 (player A) are consistent with Figure 6.

158. Additionally, for the same reasons described above regarding the view-generating logic, a POSITA would understand that the controller logic on each computer system would be the same. For example, it would be obvious to a POSITA that each computer would process “purely local events” in the same way, such as local events for setting display options.¹³⁷ Again, any POSITA would, whenever technically feasible, seek to avoid the expense and inefficiency of developing different logic and program code to perform the same task, as a matter of both common sense and sound software engineering practice.

159. This description meets the limitation under both proposed constructions in the Related Litigation because the controller logic program code causes the model to be changed **and** the view to be refreshed, not just one or the other. Accordingly, Roy discloses each limitation in (b). I discuss limitation (c) next.

	Claim 1	Claim 13	Claim 25	Claim 26
(c)	wherein one or more portions of	wherein one or more portions of the	wherein one or more portions of	wherein one or more portions of

¹³⁶ Roy at Fig. 5A and 5:17–49 (highlighting added) (Ex. 1004).

¹³⁷ Roy; 5:40–43 (Ex. 1004).

	Claim 1	Claim 13	Claim 25	Claim 26
	the application are performed at the client without the client having to interact with the server, and	application are performed at the client without the client having to interact with the server such that interactions between the client and server are reduced, and	the application are performed at the client without the client having to interact with the server, and	the application are performed at the client without the client having to interact with the server, and

160. In my opinion, as described above regarding the independent claims' preamble, Roy discloses wherein one or more portions of the application (e.g., the online application) are performed at the client (e.g., [computer system 60](#)) without the client having to interact with the server (e.g., [computer system 62](#) in Figure 6). Thus, interactions between client and server are necessarily reduced as required by the preamble. I incorporate those explanations here.

161. Roy further discloses embodiments wherein certain events (those that are “purely local”) are processed only on the client system. For example, in Roy Figure 8A, if the step 803 “event is a local event (i.e., if the event is intended only for use on the local machine), then the slave event server sends the event only to its local database (step 804):”¹³⁸

¹³⁸ Roy at Fig. 8A (annotated, highlighting added) and 6:46–67 (Ex. 1004).

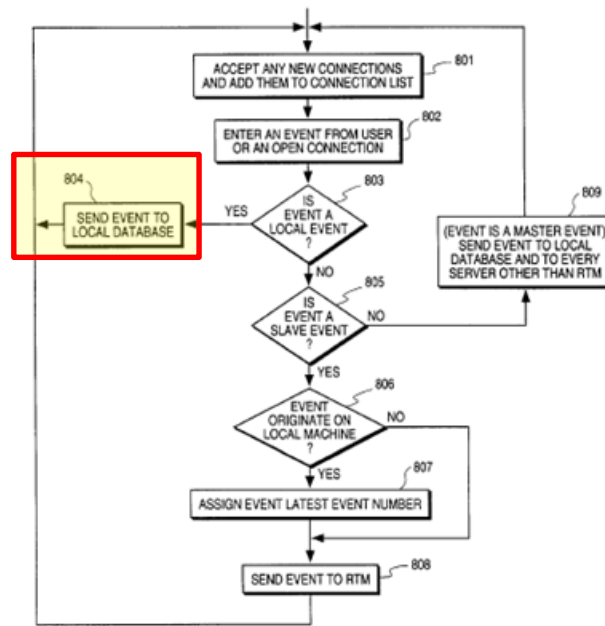


FIG. 8A

162. “Local events” are processed at the client without having to interact with the server.¹³⁹ Accordingly, Roy discloses each limitation of independent claims (c). I discuss limitation (d) next.

	Claim 1	Claim 13	Claim 25	Claim 26
(d)	further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.	further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.	further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.	further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith;

¹³⁹ Roy; Figure 8A and 5:29-46 (Ex. 1004).

163. I have already explained how, in my opinion, Roy discloses that “the client and the server both locally maintain at least a portion of the model” (e.g., a portion of the application model), and execute the view-generating and controller logic associated therewith, as described above regarding claim elements (a) and (b). I incorporate those explanations here.

164. I have also explained how in the model, view-generating logic, and controller logic for claim elements (a) and (b): (1) both the [client](#) and [server](#) “maintain at least a portion of the model” (e.g., [database 68](#) and [database 72](#)), (2) the [client](#) and [server](#) “execute the view-generating . . . logic” (e.g., [rendering commands 69 updating display 70](#) and [rendering commands 73 updating display 74](#)), and (3) the [client](#) and [server](#) “execute the . . . controller logic” (e.g., [event server 65 processing events 63](#) and [event server 67 processing events 64](#)). I incorporate those explanations here. Accordingly, Roy discloses limitation (d). I discuss limitation (e) next.

Claim 26	
(e)	further wherein, in accordance with such a dual model-view-controller arrangement, a model comprises application data, rules, and algorithms affecting the data, a view comprises a screen or window representation of a subset of the model that the application chooses to display, and a controller comprises the logic that processes user requests, and causes the model to be changed and the view to be refreshed.

165. In my opinion, Roy discloses “a model [that] comprises application data, rules, and algorithms affecting the data” (e.g., [database 68](#) and [database 72](#)). Roy discloses “a view [that] comprises a screen or window representation of the subset of the model that the application chooses to display” (e.g., [display 70](#) updated by [rendering commands 69](#) and [display 74](#) updated by [rendering commands 73](#)). Roy discloses “a controller [that] comprises the logic and processes user requests, and causes the model to be changed and the view to be refreshed” (e.g., [event server 65](#) handling [events 63](#) and [event server 67](#) handling [events 64](#)). I discussed each of these limitations above regarding the preamble, limitation (a) and limitation (b). I incorporate those explanations

here. Accordingly, Roy discloses limitation (e), and thus discloses every limitation of the independent claims of the '719 Patent.

B. Roy Renders Obvious Independent Claims 1, 13, 25, and 26

166. For the reasons stated above in §VII.A and §VIII.A, Roy also renders obvious the independent claims of the '719 Patent.

167. I understand that per agreed claim constructions in the Related Litigation, “the client and the server both locally . . . execute the view-generating and controller logic” means that the “view-generating logic” and “controller logic” executed on client must be the same “view-generating logic” and “controller logic” executed on the server, or some subset thereof. A POSITA would understand that Roy’s “view-generating logic” and “controller logic” are the same whether on the server or client, as any POSITA would, whenever technically feasible, seek to avoid the expense and inefficiency of developing different logic and program code to perform the same task, as a matter of both common sense and sound software engineering practice.

168. Accordingly, to the extent Roy does not anticipate the independent claims of the '719 Patent, it renders them obvious.

C. Roy, in Combination with Fennell and O'Reilly, Renders Obvious Dependent Claims 2, 4–12, 14, and 16–24

169. As discussed above, Roy anticipates or renders obvious each independent claim of the '719 Patent. Roy, in combination with Fennell, further in combination with O'Reilly, renders obvious dependent claims 2, 4–12, 14, and 16–24. A POSITA would have been motivated to combine Roy, Fennell, and O'Reilly as described below.

(1) Motivation to Combine

a. A POSITA Would Have Been Motivated to Combine Roy and Fennell

170. A POSITA would have been motivated to modify the online application Roy describes based on Fennell’s teachings about browser-enabled applications. Both Roy and Fennell are concerned with managing network latency effects in online systems and minimizing server interaction.

Roy at Abstract (Ex. 1004)	Fennell, at 1:39-43 (Ex. 1005)
“A method and apparatus for reducing network latency during execution of a multiplayer game across a computer network are provided.”	“This invention relates to a method of managing user inputs and displaying outputs in a multi-player game that is played on a plurality of terminals over a network, in a manner that reduces or eliminates the effects or differences in network latency among terminals.”

171. The Fennell invention related to a method of managing user inputs and displaying outputs in an application “in a manner that reduces or eliminates the effects of differences in network latency among terminals.”¹⁴⁰ Network latency effects could make a game “less enjoyable and therefore less popular,” and thus Fennell’s invention was to minimize the effects of variation of network latency.¹⁴¹

172. Roy likewise identifies that latencies associated with the Internet can be substantial and quite unpredictable.¹⁴²

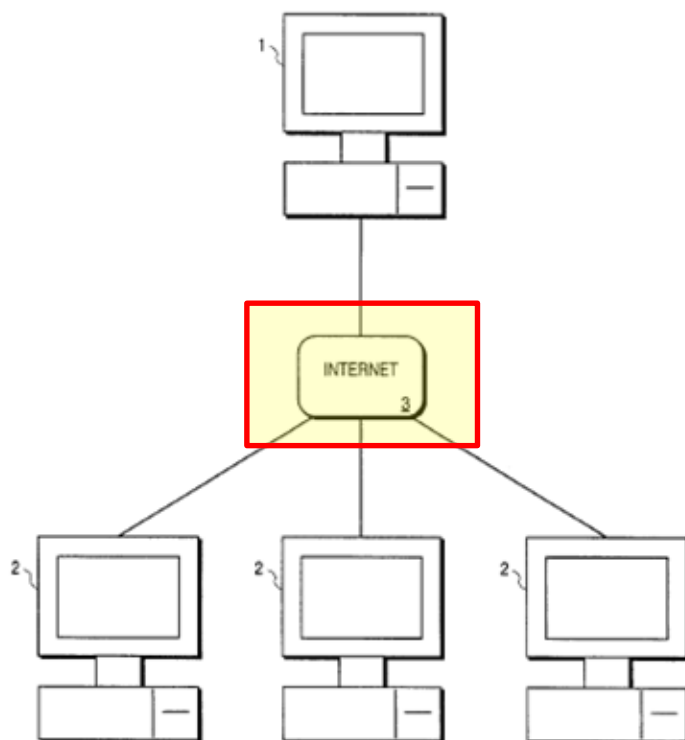
173. Figure 1 of Roy demonstrates that the network was connected via the Internet:¹⁴³

¹⁴⁰ Fennell (Ex. 1005) at 1:39–43; 3:1–2.

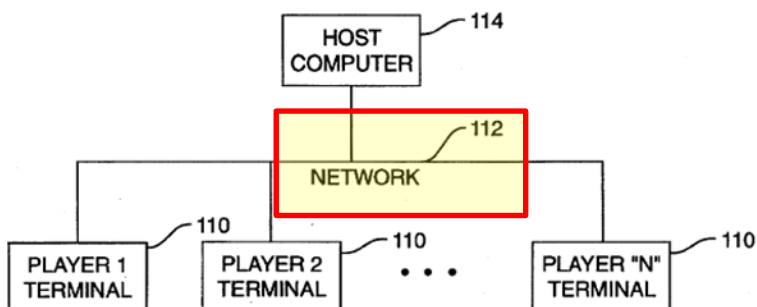
¹⁴¹ Fennell (Ex. 1005) at 1:25–36.

¹⁴² Roy at 1:49–63 (Ex. 1004).

¹⁴³ Roy at Fig. 1 (highlighting added) (Ex. 1004).



174. Figure 1 of Fennell similarly depicts a network, and explains that “network” includes “the Internet.”¹⁴⁴



¹⁴⁴ Fennell at Fig. 1 (highlighting added) and 3:23-24 (Ex. 1005).

175. In this embodiment, the Fennell “terminal” refers to any suitable input/output device such as “an appropriately programmed computer with a keyboard and display; a personal digital assistant (PDA) with a touchscreen; a telephone with a keypad and a receiver; a teletype; a television with a remote control (the well-known ‘set-top box’ concept) and the like.”¹⁴⁵ Fennell also teaches that the terminal could be “a dedicated device with programming permanently ‘burned into’ a read-only memory (ROM), a **smart terminal such as a World Wide Web browser with software ‘applets’ (written in, e.g., Sun Microsystems’ Java language) downloaded on demand, or a general-purpose computing device with suitable game programming that is conventionally loaded from a hard disk**, CD-ROM, network server, or other mass storage device when the game is to be played.”¹⁴⁶ Indeed, Fennell suggests that many embodiments would take the form of “a dedicated device,” or “a smart terminal such as a World Wide Web browser with software ‘applets’” downloaded on demand because of the low marginal cost of distributing software. A POSITA seeking to distribute the system Roy discloses would therefore look to Fennell’s browser-based disclosure.¹⁴⁷

176. In particular, a POSITA examining Roy’s application would need to determine how to distribute the application to each client computer, while also examining methods for reducing network latency effects, and would have understood that Fennell’s disclosure of browser-based application with browser software would work in the Roy system.¹⁴⁸

¹⁴⁵ Fennell at 3:3-8 (Ex. 1005).

¹⁴⁶ Fennell (Ex. 1005) at 3:11-19 (emphasis added).

¹⁴⁷ Fennell (Ex. 1005) at 3:19-22.

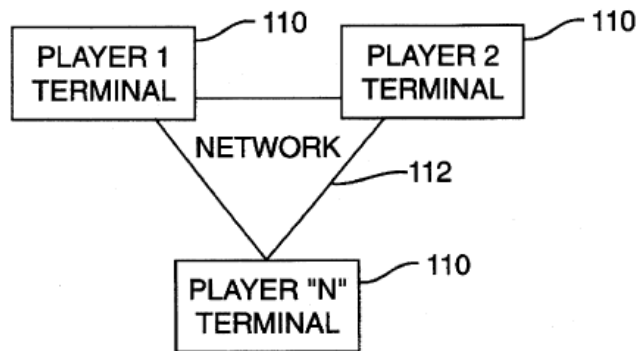
177. Further, Fennell, like Roy, discloses an embodiment wherein host computer interaction is minimized via equipping each user with a model (game challenge), a view (display) and controller (terminal equipped with an input/output device):

[T]erminals 110 may be programmed so that one of the terminals 110 transmits the game challenges. Such embodiments would permit players to manage their own games without the need for active management by the host computer 114. For example, in a trivia game, at the beginning of the game the host computer 114 may download an encrypted question-and-answer file to the terminal 110 of a particular player (e.g., a player who initiates the game by issuing an electronic-mail or “chat” challenge to other players). This player at this terminal may issue a game challenge by pressing an appropriate function key on his or her terminal 110, whereupon that player's terminal 110 both transmits the game challenge over the network 112 to the other terminals and displays the game challenge.¹⁴⁹

178. In that embodiment, host computer interaction is minimized as terminal 110 downloads an encrypted question-and-answer file (the model) and causes that model to be sent to additional participants. The additional participants do not interact with the host computer, but only with terminal 110, via network 112. Each additional participant would therefore have a copy of the model, a display (view) and controller (terminal equipped with “any suitable input/output device”¹⁵⁰):

¹⁴⁹ Fennell at 5:38–53. (Ex. 1005).

¹⁵⁰ Fennell at 3:3–4 (Ex. 1005).



151

179. Like the '719 Patent and Roy, Fennell discloses a distributed system with each terminal having a model or subset thereof, view, and controller.

180. Both Fennell and Roy address challenges associated with minimizing latency effects in multi-user environments. Among other techniques, Roy reduces latency by processing local events locally without client/server interaction. Fennell addresses the synchronization of user actions in multiuser environments by assigning ranks to each user, and further eliminating host computer interaction in peer-to-peer embodiments. A POSITA would recognize the synergy between these technologies to enhance the overall performance and user experience of networked multiuser applications.

181. Both patents are in the same technical field (networked multi-user systems) and their combination would have been predictable to a POSITA. Applying the latency reduction methods from Roy to the synchronization and host-computer-interaction-elimination method of Fennell would yield a predictably improved more user-friendly application. A POSITA would naturally consider Fennell's disclosure of a browser-enabled application to the Roy system as Fennell makes clear that browser-enabled applications are a preferred method for accessing the

¹⁵¹ Fennell at Fig. 3 (Ex. 1005).

system. What would result is a browser-enabled system where client/server interactions are minimized as each client in the system has a copy of the model, view generating logic, and controller logic.

182. Accordingly, a POSITA seeking to distribute and develop an online application with reduced network latency effects and minimized client/server interactions would examine both Roy and Fennell and understand they could deploy the Internet-based application both references describe over terminals/computers enabled with browser software.

**b. A POSITA Would Have Been Motivated to
Combine Roy and O'Reilly**

183. A POSITA would have been motivated to combine Roy's latency-reducing distributed online system with O'Reilly, which explains how Java and JavaScript can interact in both web browsers and web servers.

184. Roy focuses on reducing latency in multiuser environments, a crucial requirement for real-time data synchronization in multiuser applications. O'Reilly discloses the use of JavaScript, providing an event-driven and highly interactive scripting environment. Both address challenges in creating dynamic, user-interactive systems that rely on efficient data handling and synchronization.

185. During the time of these inventions, there was a growing trend toward building interactive web applications that required both fast data synchronization and dynamic, user-responsive interfaces. A POSITA would naturally consider integrating a proven latency-reduction mechanism (from Roy) with O'Reilly's capabilities to meet these design goals. By implementing Roy's database synchronization methods with JavaScript-driven client interfaces, developers could ensure that data updates were reflected rapidly across user devices, minimizing the perception of latency in shared applications.

186. A POSITA developing an architecture for an online system would likely have consulted O'Reilly, a readily available handbook, as it provided information, examples, and practical solutions for the problems associated with designing such systems. The '719 Patent, Roy, and O'Reilly all shared the same field of endeavor—ways of efficient information handling over the Internet.

187. O'Reilly teaches that one benefit of JavaScript is that developers can deploy it in ways that reduce server and client interaction.

188. For example, O'Reilly explains that a tax calculator could be developed with JavaScript without requiring the server “to perform a computation every time the user entered a value or clicked on a button.” JavaScript could instead conduct local computations.¹⁵² Similarly, O'Reilly explains how using JavaScript eliminates round-trips to the server when pre-processing data in user forms: “Client-side JavaScript can also perform preprocessing of input data, which can reduce the amount of data that must be transmitted to the server.”¹⁵³

189. Roy, likewise, teaches a system wherein client/server interactions were reduced, as discussed above in connection with the independent claim preamble, limitation (a), and limitation (b), which is incorporated herein.

190. A POSITA would find it obvious to combine Roy's latency-reduction techniques with the interactive capabilities of JavaScript O'Reilly describes to create a low-latency, real-time, user-interactive web application. This combination addresses complementary technical challenges and aligns with known design trends in web development, leading to predictable improvements in performance and user experience.

¹⁵² O'Reilly (Ex. 1006) at 23–28 (§1.5).

¹⁵³ O'Reilly (Ex. 1006) at 20 (§1.3.3).

**c. A POSITA Would Have Been Motivated to
Combine Roy, Fennell, and O'Reilly**

191. As discussed, a POSITA would have been motivated to combine the teachings of Roy with Fennell. A POSITA would have been further motivated to combine the resultant browser-enabled multiuser Internet application with O'Reilly for at least three reasons.

192. First, as the '719 Patent acknowledges was known in the art, "JavaScript can be used to implement much of the same functionality that applets provide."¹⁵⁴ Fennell discloses browser-based Java applets, specifically stating that in its system, a terminal could be "a smart terminal such as a World Wide Web browser with software 'applets' (written in, e.g., Sun Microsystems' Java language) downloaded on demand."¹⁵⁵ A POSITA would be motivated to combine O'Reilly with Fennell as O'Reilly describes how to use JavaScript on browsers to interact with Java applets.¹⁵⁶

193. Second, Roy and Fennell, like the '719 Patent, were concerned with minimizing network latency effects and reducing client/server interaction. O'Reilly describes how JavaScript can help eliminate client/server interaction by providing locally-updating logic in, for example, its example of a tax calculator employing JavaScript to conduct local computations¹⁵⁷ or by pre-processing data in user forms.¹⁵⁸

194. Third, an Internet-based application like Roy and Fennell describe is necessarily dynamic, and as I described above in § VII.C, JavaScript was developed out of a need and desire

¹⁵⁴ '719 Patent (Ex. 1001) at 3:1–13; 2:22–23.

¹⁵⁵ Fennell (Ex. 1005) at 3:11–19 (emphasis added).

¹⁵⁶ O'Reilly (Ex. 1006) at 283–312 (§19) (discussing the interoperability between JavaScript and Java).

¹⁵⁷ O'Reilly (Ex. 1006) at 23–28 (§1.5).

¹⁵⁸ O'Reilly (Ex. 1006) at 20 (§1.3.3).

to create dynamic behavior on HTTP web pages. A POSITA would have likely consulted O'Reilly to learn how to enable dynamic web content.

195. Accordingly, a POSITA would have had considerable motivation to combine the teachings of Roy with Fennell (as discussed above) further with O'Reilly.

(2) Claims 2 and 14: “a HyperText Transport Protocol”

196. Fennell teaches that an online Internet-based application like Roy can be deployed in conjunction with a browser. Further, a POSITA seeking to deploy a browser-based application in the relevant timeframe would have strong motivation to consult O'Reilly, as it describes how to use JavaScript to generate HTML documents for the browser. O'Reilly—like the '719 Patent itself—confirms that a POSITA would be familiar with HTTP and HTML in connection with network communications by no later than 1997.

Claim 2	Claim 14
The method of claim 1, wherein the client and server communicate over a HyperText Transport Protocol network.	The system of claim 13, wherein the network is a HyperText Transport Protocol network.

197. Claims 2 and 14 require a HyperText Transfer Protocol network (HTTP). This is disclosed in prior art cited in the '719 Patent prosecution history, '719 Patent, and O'Reilly.

198. First, I understand that the Examiner during prosecution took “Official Notice” that “HTTP is well-known in the art as taught by Gish [U.S. Patent No. 5,768,510], wherein Gish teaches the client and server communicate over a HyperText Transport Protocol network[.]”¹⁵⁹

¹⁵⁹ File history of U.S. Patent No. 7,702,719 (Application No. 09/500,208) at August 9, 2006 Office Action at 316–317 (Ex. 1002).

199. Second, the '719 Patent acknowledges that HTTP was well-known in the art: “The network 3000 may employ a HyperText Transport Protocol (HTTP), **as is well known in the art and as is implemented in accordance with the world wide web[.]**”¹⁶⁰ A POSITA from the relevant time would understand that *any* web-based client/server system could use HTTP for communication.

200. Finally, the combination of Roy with browser-based disclosures of Fennell combined with O'Reilly render this limitation obvious. Throughout the book, O'Reilly presupposes the readers' familiarity with HTTP. For example, O'Reilly teaches that “Cookies help you provide the state information that is missing from the stateless HTTP protocol of the Web.”¹⁶¹ When discussing implementations of Server-Side JavaScript, O'Reilly explains how to handle certain HTTP requests.¹⁶² O'Reilly discusses implementation of cookies as “an extension to the HTTP protocol.”¹⁶³ In short, O'Reilly presupposes familiarity with HTTP because it was well known. I routinely taught HTTP to computer science undergraduates at least as early as 1995.

201. In my opinion, claims 2 and 14 are thus rendered obvious.

(3) Claims 4 and 16: “partition/ing a screen area associated with the browser software into frames”

202. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent, and together with Fennell, renders obvious claims 3 and 15. Roy, in combination with Fennell and O'Reilly, renders obvious dependent claims 4 and 16.

¹⁶⁰ '719 Patent at 4:58–60.

¹⁶¹ O'Reilly (Ex. 1006) at 21 (§1.3.5).

¹⁶² O'Reilly (Ex. 1006) at 32 (§1.6.5).

¹⁶³ O'Reilly (Ex. 1006) at 234 (§15.1).

Claim 4	Claim 16
The method of claim 3, wherein the configuring step further comprises the step of partitioning a screen area associated with the browser software into frames. ¹⁶⁴	The system of claim 15, wherein the client processor is further operative to partition a screen area associated with the browser software into frames.

203. A POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell’s teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

204. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O’Reilly, which discloses programming techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

205. In my opinion, Roy, in combination with Fennell and O’Reilly renders obvious claims 4 and 14 of the ’719 Patent. O’Reilly expressly discloses partitioning a screen area associated with browser software into frames, which renders obvious the limitations of claims 4 and 16, if claim 4 refers to the “configuring step” (b).¹⁶⁵

206. For example, O’Reilly discloses a browser window that is capable of partitioning a screen area associated with the browser into frames:

While Navigator (and other browsers) can display multiple HTML pages in multiple browser windows, it can also display multiple pages within a single window by using frames—a feature that allows a single browser window to be divided horizontally or

¹⁶⁴ As noted above, it is not clear which “configuring step” of claim 1 claim 4 refers to, whether it relates to (a) “configuring the server to store a model associated with the application and to execute view- generating and controller logic associated with the application” or (b) “configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application”.

¹⁶⁵ This is true whether a frame is construed as a “section of a webpage, each constituting a distinct HTML document” or “plain and ordinary meaning.”

vertically (or both) into individual sections that each display a separate HTML document. Although frames are not strictly windows in their own right, they behave like windows in many ways, and in JavaScript, each frame is represented with a Window object. Thus the Window class can represent both top-level browser windows and frames within a browser window (and frames within other frames, of course).¹⁶⁶

207. A POSITA would understand that to enable this feature, a system would necessarily require a client processor that is operative to partition a screen area associated with the browser software into frames, as relevant to claim 16.

208. If the configuring step of claim 4 refers to independent claim limitation (a), as opposed to (b), O'Reilly still discloses the limitation. O'Reilly discloses that server-side JavaScript can dynamically generate HTML to be displayed by the client, which would necessarily be enabled to display that information in frames as noted above:

Server-side JavaScript provides an alternative to CGI scripts. It goes beyond the CGI model, in fact, because server-side JavaScript is embedded directly within HTML pages and allows executable server-side scripts to be directly intermixed with web content. Whenever a document containing server-side JavaScript code is requested by the client, the server executes the script or scripts contained in the document and sends the resulting document (which may be partially static and partially dynamically generated) to the requester. Because execution speed is a very important issue on production web servers, HTML files that contain server-side JavaScript are precompiled to a binary form that may be more efficiently interpreted and sent to the requesting client. An obvious capability of server-side JavaScript is to dynamically generate HTML to be displayed by the client.¹⁶⁷

209. Accordingly, a POSITA seeking to distribute and develop an Internet-based browser-enabled application as Roy and Fennell disclose would understand that partitioning a screen area associated with the browser software into frames was a known capability. Thus Roy, in combination with Fennell and O'Reilly renders claims 4 and 16 of the '719 Patent obvious.

¹⁶⁶ O'Reilly (Ex. 1006) at 184 (§11.3) (highlighting added).

¹⁶⁷ O'Reilly (Ex. 1006) at 32 (§1.6.5) (highlighting added).

(4) Claims 5 and 17: “wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are associated with at least one frame and one or more views for display in accordance with the application are associated with at least another frame”

210. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent, and together with Fennell renders obvious claims 3 and 15, and further in combination with O'Reilly renders obvious claims 4 and 16. Roy, in combination with Fennell and O'Reilly renders obvious dependent claims 5 and 17.

Claim 5	Claim 17
The method of claim 4, wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are associated with at least one frame and one or more views for display in accordance with the application are associated with at least another frame.	The system of claim 16, wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are associated with at least one frame and one or more views for display in accordance with the application are associated with at least another frame.

211. A POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell's teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

212. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O'Reilly, which provides programming techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

213. As discussed above in connection with the independent claim preamble and limitations (a) and (b), Roy discloses a “dual-MVC” approach that places a copy of the model, view-generating logic and controller logic on a client. (see § VII.A). A POSITA would find it

obvious, based on the teachings of O'Reilly, to associate the model, view-generating logic, and controller logic with a frame that is separate from the viewed frame.

214. First, O'Reilly discloses that JavaScript can be used to associate a subset of a model associated with an application with a frame.¹⁶⁸ O'Reilly teaches using JavaScript to associate tax bracket information with a frame.¹⁶⁹

215. Second, O'Reilly discloses that JavaScript can be used to associate view-generating logic with a frame, and expressly states that “one commonly used feature of JavaScript is the ability to dynamically generate HTML and JavaScript content for display in other browser windows and frames.”¹⁷⁰ In this example, the JavaScript code creating a dynamic HTML document is view-generating logic, because the HTML document that is displayed is a result of the JavaScript coding specifying the content and look of the HTML document.

216. Third, O'Reilly discloses that JavaScript can be used to associate controller logic with a frame. Specifically, O'Reilly teaches frame programming techniques that deploy multiple frames, at least one of which can run a function and at least another of which can dynamically display the results of that function.¹⁷¹

Once you know how to refer to any frame from any other frame, you can start writing JavaScript programs that work in complex framed documents. Pay careful attention to how you name frames, and be aware of what window any given piece of code is running in. For example, if an event handler in frame A invokes a function that is defined in frame B, the code in that function is running in frame A, not frame B—and if the code wants to refer to frame B, it can't just use the implicit window reference, as it could if it were actually running in frame B.

¹⁶⁸ O'Reilly (Ex. 1006) at 208 (§12.4).

¹⁶⁹ O'Reilly (Ex. 1006) at 23–28 (§1.5); 207–09 (§12.4).

¹⁷⁰ O'Reilly (Ex. 1006) at 164 (§10.1.2, FN†).

¹⁷¹ O'Reilly (Ex. 1006) at 208 (§12.4) (highlighting added).

217. In this example, the “event handler” in frame A is controller logic, because O’Reilly defines “event handlers” as “pieces of JavaScript code that are executed when a particular event occurs, such as when the user clicks on a button.”¹⁷²

218. Fourth, O’Reilly teaches that the frame associated with the model, view-generating logic, and controller logic should preferably be associated with at least one frame that is different from the frame associated with a view. O’Reilly teaches that it is actually *harder* for JavaScript to create a view in the same frame associated with the logic to create the view, because doing so would overwrite the JavaScript code itself. Instead, O’Reilly teaches it is better to separate the display into its own frame for dynamic updates:¹⁷³

As we saw in the error handler example, JavaScript code in one window (or frame) can dynamically create an HTML document in another window (or frame). It is a lot harder for JavaScript code to dynamically create a new HTML document in its own window or frame, because doing this generally overwrites the JavaScript code itself! If your web page design calls for one static frame and two frames that have their contents dynamically updated, the static frame can contain the JavaScript code necessary to update the dynamic frames. But what if your design calls for all the frames to be dynamic? A static frame is still required, but the trick here is to create the static frame so that it is invisible! You do this by explicitly creating it at a location that is greater than 100% of the frame width or height. HTML to create such an invisible frame is shown in Example 12.5.

219. O’Reilly discloses an example of generating dynamic HTML content wherein the frames themselves are dynamically created.¹⁷⁴

We’d described how you can use JavaScript running in one frame to dynamically create HTML content for another frame. But in this discussion we have always created the frames themselves with a frameset specified in a static HTML file. Since frames are specified in HTML, there is no reason we cannot create them dynamically as well. Example 12.6 shows how it can be done. This example opens a small new window, dynamically creates four frames in it, and then, using the `setTimeout()` method, periodically changes the background color of each frame, creating a simple but colorful animated display, which is pictured in Figure 12.3. The Stop button in the original

¹⁷² O’Reilly (Ex. 1006) at 15 (§1.1).

¹⁷³ O’Reilly (Ex. 1006) at 208 (§12.4) (highlighting added).

¹⁷⁴ O’Reilly (Ex. 1006) at 209 (§12.4) (highlighting added).

window stops the animation using `clearTimeout()` and closes the new window using the `Window.close()` method.

220. Additionally, O'Reilly specifies that the “`write()`” method can dynamically generate an HTML document that is displayed as intended, noting that it would be preferable to do so from one frame into another frame:¹⁷⁵

Without a doubt, the most important feature of the Document object (and perhaps of client-side JavaScript in general) is the `write()` method, which allows us to dynamically generate web page content from our JavaScript programs. There are several ways that this method can be used. The most obvious is to use it within a script to output HTML into the document that is currently being parsed. This is the way it was used above to display the Document `lastModified` property at the bottom of the web page. Be aware that you can only output HTML to the current document while that document is being parsed. That is, you can only call `document.write()` from within `<SCRIPT>` tags, because these scripts are executed as part of the document parsing process. In particular, if you call `document.write()` from an event handler, you will end up overwriting the current document (including its event handlers), instead of appending text to it. Although you can't usefully write to the current document from an event handler, there is no reason you can't write to a document in another window or frame, and doing so can be a very useful technique for multiwindow or multiframe web sites.

221. O'Reilly therefore teaches it is better to have the “view” associated with a different frame from the frame that creates the view. Otherwise, O'Reilly teaches, the application code creating the HTML document would be overwritten.

222. Frames were a well-known tool for isolating functionality in browser-based systems, as O'Reilly teaches. Associating different application components (controller logic vs. view generating logic) within separate frames would have been a predictable implementation for a modular, browser-based client-server system the combination of Roy and Fennell disclose.

223. Accordingly, a POSITA would understand that to deploy the Internet-based browser-enabled application Roy and Fennell teach, it would be beneficial to separately associate

¹⁷⁵ O'Reilly (Ex. 1006) at 224 (§14.1.2) (highlighting added).

the subset of the model, view-generating and controller logic associated with the application with one frame, and the resultant views for display with another frame, as O'Reilly teaches.

224. Thus, in my opinion, Roy in combination with Fennell and O'Reilly renders obvious claims 5 and 17 of the '719 Patent.

(5) Claims 6 and 18: “wherein the at least one view frame is a visible frame”

225. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent, and together with Fennell renders obvious claims 3 and 15, and further in combination with O'Reilly renders obvious claims 4, 5, 16, and 17. Roy, in combination with Fennell and O'Reilly renders obvious dependent claims 6 and 18.

Claim 6	Claim 18
The method of claim 5, wherein the at least one view frame is a visible frame.	The system of claim 17, wherein the at least one view frame is a visible frame.

226. A POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell's teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

227. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O'Reilly, which provides programming techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

228. A POSITA would understand that in order to have a workable Internet-based browser-enabled application, the view of the application would necessarily need to be visible to the user. For example, O'Reilly discloses a browser window that is capable of partitioning a screen area associated with the browser into frames, and **displaying** HTML documents:

While Navigator (and other browsers) can display multiple HTML pages in multiple browser windows, it can also display multiple pages within a single window by using frames--a feature that allows a single browser window to be divided horizontally or vertically (or both) into individual sections that each display a separate HTML document.¹⁷⁶

229. A POSITA would understand that in order for a display to be perceived by a user, it would need to be visible. O'Reilly teaches frame programming techniques that deploy multiple frames, at least one of which can run a function and at least another of which can dynamically display the results of that function.¹⁷⁷

Once you know how to refer to any frame from any other frame, you can start writing JavaScript programs that work in complex framed documents. Pay careful attention to how you name frames, and be aware of what window any given piece of code is running in. For example, if an event handler in frame A invokes a function that is defined in frame B, the code in that function is running in frame A, not frame B—and if the code wants to refer to frame B, it can't just use the implicit window reference, as it could if it were actually running in frame B.

230. O'Reilly expressly teaches that it is beneficial to separate the display into its own frame for dynamic updates, and distinguished visible and invisible frames:¹⁷⁸

As we saw in the error handler example, JavaScript code in one window (or frame) can dynamically create an HTML document in another window (or frame). It is a lot harder for JavaScript code to dynamically create a new HTML document in its own window or frame, because doing this generally overwrites the JavaScript code itself! If your web page design calls for one static frame and two frames that have their contents dynamically updated, the static frame can contain the JavaScript code necessary to update the dynamic frames. But what if your design calls for all the frames to be dynamic? A static frame is still required, but the trick here is to create the static frame so that it is invisible! You do this by explicitly creating it at a location that is greater than 100% of the frame width or height. HTML to create such an invisible frame is shown in Example 12.5.

231. Accordingly, in my opinion, Roy in combination with Fennell and O'Reilly renders obvious claims 6 and 18 of the '719 Patent.

¹⁷⁶ O'Reilly (Ex. 1006) at 184 (§11.3) (highlighting added).

¹⁷⁷ O'Reilly (Ex. 1006) at 208 (§12.4) (highlighting added).

¹⁷⁸ O'Reilly (Ex. 1006) at 207–11 (§12.4) (highlighting added).

(6) Claims 7 and 19: “wherein the at least one frame associated with the at least a subset of the model, the view-generating logic and the controller logic is not a visible frame”

232. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent, and together with Fennell renders obvious claims 3 and 15, and further in combination with O'Reilly renders obvious claims 4, 5, 16, and 17. Roy, in combination with Fennell and O'Reilly renders obvious dependent claims 7 and 19.

Claim 7	Claim 19
The method of claim 5, wherein the at least one frame associated with the at least a subset of the model, the view generating logic and the controller logic is not a visible frame.	The system of claim 17, wherein the at least one frame associated with the at least a subset of the model, the view generating logic and the controller logic is not a visible frame.

233. A POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell's teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

234. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O'Reilly, which provides programming techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

235. Specifically, O'Reilly teaches frame programming techniques that deploy multiple frames, at least one of which can run a function and at least another of which can dynamically display the results of that function.¹⁷⁹

Once you know how to refer to any frame from any other frame, you can start writing JavaScript programs that work in complex framed documents. Pay careful attention to how you name frames, and be aware of what window any given piece of code is running in. For example, if an event handler in frame A invokes a function that is defined in

¹⁷⁹ O'Reilly (Ex. 1006) at 208 (§12.4) (highlighting added).

frame B, the code in that function is running in frame A, not frame B—and if the code wants to refer to frame B, it can't just use the implicit window reference, as it could if it were actually running in frame B.

236. O'Reilly further teaches that it is actually *harder* for JavaScript to create a view in the same frame associated with the logic to create the view, and thus recommends that a programmer use an *invisible* frame:¹⁸⁰

As we saw in the error handler example, JavaScript code in one window (or frame) can dynamically create an HTML document in another window (or frame). It is a lot harder for JavaScript code to dynamically create a new HTML document in its own window or frame, because doing this generally overwrites the JavaScript code itself! If your web page design calls for one static frame and two frames that have their contents dynamically updated, the static frame can contain the JavaScript code necessary to update the dynamic frames. But what if your design calls for all the frames to be dynamic? A static frame is still required, but the trick here is to create the static frame so that it is invisible! You do this by explicitly creating it at a location that is greater than 100% of the frame width or height. HTML to create such an invisible frame is shown in Example 12.5.

237. O'Reilly explains how to create an invisible frame:¹⁸¹

Example 12.5: Creating an Invisible Frame

```
<!-- Create two frames that take up half the screen each, and one that
-->
<!-- takes up "all the rest" of the room. The third frame will be -->
<!-- invisible, because it has a height of zero. -->
<frameset rows="50%,50%,*">
<!-- first two frames start out empty, loading no documents -->
<frame name="dynamic_frame_1">
<frame name="dynamic_frame_2">
<!-- invisible frame contains the code that will -->
<!-- dynamically update the others -->
<frame name="invisible_frame" src="program.html">
</frameset>
```

238. In this example, the third “invisible” frame is defined in the frameset as “*,” and will not be visible because it has a height of zero. The invisible frame contains a reference to a separate JavaScript program (“program.html”) that can dynamically update the visible frames.

¹⁸⁰ O'Reilly (Ex. 1006) at 208 (§12.4) (highlighting added).

¹⁸¹ O'Reilly (Ex. 1006) at 208–09 (§12.4; Example 12.5) (highlighting added).

239. As discussed above in connection with the independent claim preamble and limitations (a) and (b), Roy discloses a “dual-MVC” approach that associates a copy of the model, view-generating logic and the controller logic with a client. (see § VII.A). A POSITA would understand that to avoid overwriting the subset of the model, view-generating logic and controller logic, it would be beneficial to associate those parts of the program with an invisible frame. Otherwise, the frame could be overwritten and would lose the ability to run the application Roy discloses.

240. Accordingly, it would be obvious to a POSITA to deploy the Internet-based browser-enabled application Roy and Fennell teach, by associating at least one frame associated with the at least a subset of the model, the view generating logic and the controller logic with a not visible frame, as O’Reilly teaches.

241. Thus, in my opinion, Roy in combination with Fennell and O’Reilly renders obvious claims 7 and 19 of the ’719 Patent.

(7) Claims 8 and 20: “form/ing at least one frame with which application-independent view-generating logic and controller logic is associated”

242. As discussed above, Roy anticipates or renders obvious each of the independent claims of the ’719 Patent, and together with Fennell renders obvious claims 3 and 15, and further in combination with O’Reilly renders obvious claims 4, and 16. Roy, in combination with Fennell and O’Reilly renders obvious dependent claims 8 and 20.

Claim 8	Claim 20
The method of claim 4, wherein the configuring step further comprises forming at least one frame with which application-	The system of claim 16, wherein the client processor is further operative to form at least one frame with which application-independent view-generating logic and controller logic is associated.

Claim 8	Claim 20
independent view-generating logic and controller logic is associated. ¹⁸²	

243. First, as I described above, what the '719 Patent characterizes as “application-independent controller logic” (“generic user-interface event handling”) and “application-independent view-generating logic” (“logic to generate a button on the screen”) were well-known, widely practiced, and routinely taught in undergraduate classrooms years before the '719 Patent was filed.

244. Second, a POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell’s teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

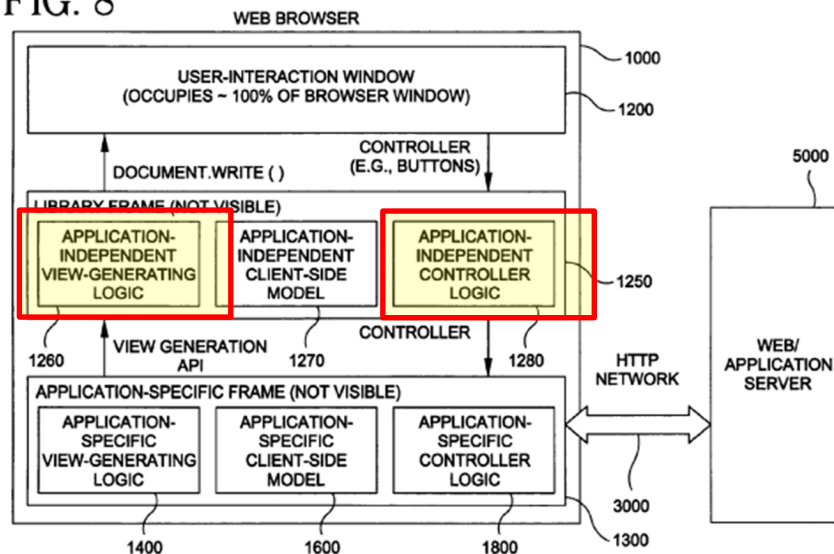
245. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O’Reilly, which provides programming techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

246. The '719 Patent discloses “application-independent” view-generating logic (1260) and controller logic (1280) in Fig. 8:¹⁸³

¹⁸² As noted above, it is not clear to which “configuring step” of claim 1 claim 4 (and by further dependence, claim 8) refers, whether it relates to (a) “configuring the server to store a model associated with the application and to execute view- generating and controller logic associated with the application” or (b) “configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application.” As the '719 Patent does not describe application-independent view-generating and controller logic with respect to configuring the server (independent claim limitation (a)), my focus will be on the independent claim limitation (b).

¹⁸³ '719 Patent (Ex. 1001) at Fig. 8 (highlighting added).

FIG. 8



247. The '719 Patent does not further describe the “application-independent” view-generating logic. The '719 Patent describes library frame 1250 and application independent controller logic 1280 as follows:¹⁸⁴

The library frame 1250 provides an application-programming-interface (API) and implementation for generating the view in the visible frame. The API allows the programmer to specify the layout in terms of visible components, with each component having controller properties, e.g., text values, color. The implementation of the components in the library writes the appropriate information into the visible frame (e.g., HTML) to generate the described visual layout. The library frame also contains an application-independent client-side model function 1270, e.g., caching and data-movement operations. Finally, the library frame contains application independent controller logic 1280, e.g., generic user-interface event handling.

248. In my opinion, Roy discloses “application-independent view-generating logic and controller logic.” Roy discloses that there are certain set display options that are “purely local” and that relate to, for example, “events which set display options or other functions that are only relevant to” a particular computer system.¹⁸⁵

¹⁸⁴ '719 Patent (Ex. 1001) at 7:51–63 (highlighting added).

¹⁸⁵ Roy (Ex. 1004) at 5:17–49.

249. Roy's disclosure of "purely local events" that "set display options or other functions" could include application-agnostic view-generating logic like setting the background color or text color of the display. These events—e.g., mouse clicks setting background color or text color—would be processed by purely local controller logic. Accordingly, a POSITA would understand that Roy discloses application-independent view-generating and controller logic.

250. In my opinion, O'Reilly discloses forming "at least one frame with which application-independent view-generating logic and controller logic is associated."

251. First, O'Reilly discloses a library of view-generating logic attributes that a developer could call on to create an HTML document, such as specifying the colors that it is displayed in:¹⁸⁶

If the Window object, which represents a window or a frame, is the central object in client-side JavaScript, then the Document object, which represents the contents of a window or frame, runs a close second, and is just as commonly used. This object has properties that specify information about the document: the URL, its last-modified date, the URL of the document that linked to it, the colors that it is displayed in. The Document object also has a few methods that allow JavaScript programs to dynamically output text into a document, and to dynamically create new documents from scratch.

252. O'Reilly further expands upon the view-related attributes:¹⁸⁷

The Document object has a number of properties that correspond to attributes of the <BODY> tag, and which are used to specify the colors that the document is displayed in. The bgColor property, and the BGCOLOR attribute specify the background color of the document. Similarly, the fgColor and the TEXT attribute specify the default color for text in the document. The linkColor property specifies the color of unvisited links, and vlinkColor and alinkColor specify the color of visited links and of activated links (i.e., links currently being clicked on). The LINK, VLINK, and ALINK attributes correspond to these properties.

¹⁸⁶ O'Reilly (Ex. 1006) at 222 (§14.1) (highlighting added).

¹⁸⁷ O'Reilly (Ex. 1006) at 222–23 (§14.1.1) (highlighting added).

253. And O'Reilly specifies that the "write()" method can dynamically generate an HTML document that is displayed as intended, and even specifies that it would be preferable to do so from one frame into another frame.¹⁸⁸

Without a doubt, the most important feature of the Document object (and perhaps of client-side JavaScript in general) is the write() method, which allows us to dynamically generate web page content from our JavaScript programs. There are several ways that this method can be used. The most obvious is to use it within a script to output HTML into the document that is currently being parsed. This is the way it was used above to display the Document lastModified property at the bottom of the web page. Be aware that you can only output HTML to the current document while that document is being parsed. That is, you can only call document.write() from within <SCRIPT> tags, because these scripts are executed as part of the document parsing process. In particular, if you call document.write() from an event handler, you will end up overwriting the current document (including its event handlers), instead of appending text to it. Although you can't usefully write to the current document from an event handler, there is no reason you can't write to a document in another window or frame, and doing so can be a very useful technique for multiwindow or multiframe web sites.

254. Accordingly, a POSITA would understand that application-independent view-generating logic would preferably be associated with a frame separate and apart from the document frame itself, and would deploy this strategy in an Internet-based browser enabled application.

255. Second, O'Reilly discloses "application-independent . . . controller logic." O'Reilly describes a number of "event handlers," which the '719 Patent states is an example of "application-independent . . . controller logic."¹⁸⁹

256. O'Reilly provides that event handlers are an important feature of JavaScript, and even provides a table of the Supported Event Handlers:¹⁹⁰

An important feature of JavaScript is the ability to define "event handlers"—arbitrary pieces of code to be executed when a particular event occurs. Usually, these events are initiated by the user, when (for example) she moves the mouse over a hypertext link or enters a value in a form or clicks the Submit button in a form. This event-handling

¹⁸⁸ O'Reilly (Ex. 1006) at 224 (§14.1.2) (highlighting added).

¹⁸⁹ '719 Patent (Ex. 1001) at 7:61–63 ("Finally, the library frame contains application independent controller logic 1280, e.g., generic user-interface event handling.").

¹⁹⁰ O'Reilly (Ex. 1006) at 21 (§1.3.4) (highlighting added); *see also* O'Reilly (Ex. 1006) at 166–69 (§10.3.2 ("Event Handlers in JavaScript"; §Table 10-1 ("JavaScript Event Handlers"))).

capability is a crucial one, because programming with graphical interfaces, such as HTML forms, inherently requires an event-driven model. JavaScript can trigger any kind of action in response to user events. Typical examples might be to display a special message in the status line when the user positions the mouse over a hypertext link, or to pop up a confirmation dialog box when the user submits an important form.

257. And O'Reilly specifies that the "write()" method can be called using event handlers, noting that it would be preferable to do so from one frame into another frame¹⁹¹:

Without a doubt, the most important feature of the Document object (and perhaps of client-side JavaScript in general) is the write() method, which allows us to dynamically generate web page content from our JavaScript programs. There are several ways that this method can be used. The most obvious is to use it within a script to output HTML into the document that is currently being parsed. This is the way it was used above to display the Document lastModified property at the bottom of the web page. Be aware that you can only output HTML to the current document while that document is being parsed. That is, you can only call document.write() from within <SCRIPT> tags, because these scripts are executed as part of the document parsing process. In particular, if you call document.write() from an event handler, you will end up overwriting the current document (including its event handlers), instead of appending text to it. Although you can't usefully write to the current document from an event handler, there is no reason you can't write to a document in another window or frame, and doing so can be a very useful technique for multiwindow or multiframe web sites.

258. In my opinion, a POSITA would understand that "application-independent . . . controller logic," like handling mouse clicks, would preferably be associated with a frame separate and apart from the document frame itself, per the teachings of O'Reilly, and would deploy this strategy in an Internet-based browser enabled application.

259. Roy, Fennell, and O'Reilly collectively emphasize modularity, enabling components to be reused across applications. Associating application-independent logic with a frame aligns with this shared objective.

260. Frames, as O'Reilly teaches, were a tool for encapsulating and isolating reusable logic known well before the '719 Patent was filed. A POSITA would recognize that associating

¹⁹¹ O'Reilly (Ex. 1006) at 224 (§14.1.2) (highlighting added).

such logic with a frame simplifies application development and maintenance, which aligns with the goals of Roy and Fennell.

261. Roy and Fennell provide methods for decentralizing logic and improving application efficiency. O'Reilly complements these teachings by providing a mechanism (frames) to encapsulate such logic, making it reusable and application-independent. Developers sought to reduce redundancy and improve scalability in client-server systems. Associating reusable, application-independent logic provides an obvious solution for achieving these objectives, enhancing both development efficiency and application performance.

262. Accordingly, a POSITA would find it obvious to have at least one frame with which application-independent view-generating logic and controller logic is associated, as O'Reilly teaches, in the Internet-based application Roy and Fennell teach.

263. Thus, in my opinion, Roy in combination with Fennell and O'Reilly renders obvious claims 8 and 20 of the '719 Patent.

(8) Claims 9 and 21: “wherein the at least one application-independent view-generating logic and controller logic frame further has an application-independent model associated therewith”

264. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent, and together with Fennell renders obvious claims 3 and 15, and further in combination with O'Reilly renders obvious claims 4, 8, 15, 16, and 20. Roy, in combination with Fennell and O'Reilly renders obvious dependent claims 9 and 21.

Claim 9	Claim 21
The method of claim 8, wherein the at least one application-independent view-generating logic and controller logic frame further has an	The system of claim 20, wherein the at least one application-independent view-generating logic and controller logic frame further has an

Claim 9	Claim 21
application-independent model associated therewith.	application-independent model associated therewith.

265. First, as I described above, what the '719 Patent characterizes as an “application-independent model”—“caching and data-movement operations”—were well-known techniques that were widely practiced and routinely taught in undergraduate classrooms decades before the '719 Patent was filed.

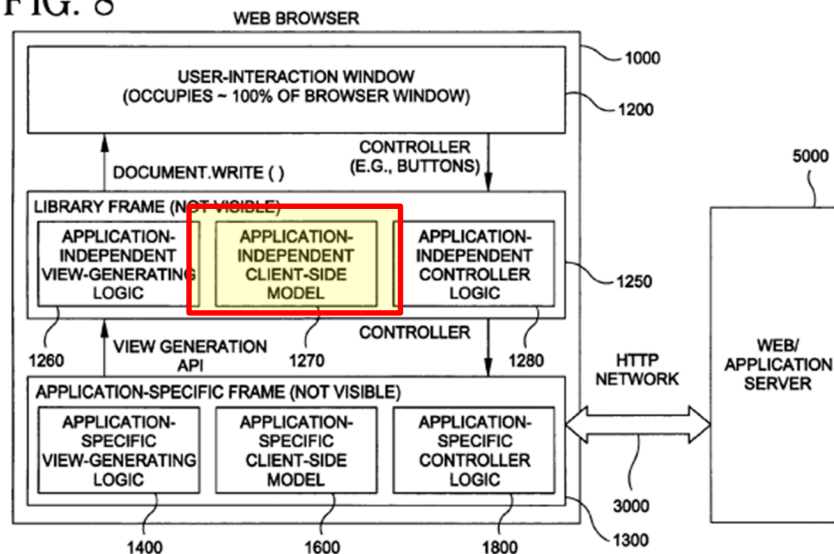
266. Second, a POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell’s teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

267. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O’Reilly, which provides programming techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

268. The '719 Patent discloses “application-independent client-side model”, shown as part of the non-visible “library frame” in Fig. 8:¹⁹²

¹⁹² '719 Patent (Ex. 1001) at Fig. 8 (highlighting added).

FIG. 8



269. The '719 Patent describes “application-independent client-side model function 1270” as, e.g., “caching and data-movement operations.”¹⁹³ No other example or description of “application-independent client side model function” appears in the '719 Patent.

270. A POSITA would already have full knowledge of caching and data-movement operations, as well as how to implement them. Indeed, caching was disclosed as early as 1965 by Maurice Wilkes, there described as a “slave memory.”¹⁹⁴ IBM itself disclosed a system employing caching and associated data movement operations in the early 1970’s in U.S. Patent No. 3,771,137 to Barner et al., “Memory Control in a Multipurpose System Utilizing a Broadcast.”¹⁹⁵ In addition, caching and associated data movement operations have been part of undergraduate and graduate computer science curricula at least as long as I have been teaching (since 1985).

¹⁹³ '719 Patent (Ex. 1001) at 7:51–63.

¹⁹⁴ Maurice Wilkes, Slave Memories and Dynamic Storage Allocation, IEEE Transactions on Electronic Computers, Vol. EC-1.4, Issue: 2, (April 1965) 270-271 (Ex. 1012).

¹⁹⁵ U.S. Patent No. 3,771,137 (Ex. 1013).

271. Because the '719 Patent states that an application independent model's function is, as an example, caching or data movement operations, Roy's disclosure of caching and data movement operations renders this limitation obvious. Roy discloses that the slave copy of the database represents cached data.¹⁹⁶

"In the preferred embodiment, the master database is located on the same machine as the master event Server. Slave copies of the master database (or portions thereof) are stored on each machine other than the machine in which the master event server is located. By providing one master copy of the database, events originating from different machines can be more easily synchronized. By providing slave copies of the database in every other machine involved in the game, the need to send rendering commands across the network is reduced or eliminated, which reduces bandwidth consumption."

272. In Roy, the slave copies of the master database are "cached" portions of the master database. The slave computer system (client) performs caching operations when it updates the slave database. Similarly, the server performs cache-related "data movement operations" when it sends events to slave computers across the network that cause the slave database (cache) to be updated. These caching and data movement operations do not depend on the specifics of the application performing them and are thus application-independent. Accordingly, a POSITA would understand that Roy discloses an application-independent model. Given the desire for browser-enabled applications as Fennell discloses, a POSITA would further be motivated to combine the teachings of Roy with O'Reilly (as described below) and Fennell to understand how to enable the application-independent model in a frame.

273. Second, O'Reilly discloses an "application-independent model." The example of an application-independent model function provided in the '719 Patent relates to caching and data movement. O'Reilly taught caching and data movement operations in JavaScript years before the '719 Patent was filed.

¹⁹⁶ Roy at 4:44–54 (Ex. 1004) (highlighting added).

274. For example, O'Reilly teaches that allowing JavaScript files to be cached by the browser allows them to load much more quickly¹⁹⁷:

When JavaScript functions are used by more than one page, placing them in a separate JavaScript file **allows them to be cached by the browser, making them load much more quickly**. When JavaScript code is shared by multiple pages, **the time savings of caching** more than outweigh the small delay required for the browser to open a separate network connection to download the JavaScript file the first time it is requested.

275. O'Reilly also teaches a method of caching images for easy replacement into dynamic HTML pages:¹⁹⁸

The ability to dynamically replace one image in a static HTML document with another image opens the door to any number of special effects, from animation, to images that change when clicked on, to "digital clocks" that update themselves in real time. With a bit of thought, you can probably imagine many more potential uses for this technique. In order to make the image replacement technique viable, and in order to make animations and other special effects responsive enough to be useful, we need some way to ensure that the necessary images are loaded into the browser's cache.

276. Further, O'Reilly teaches that caching JavaScript improves performance, noting that: "Downloading an external file of JavaScript code, even over a relatively fast modem connection, can cause noticeable delays in the loading and execution of a web page. Of course, once the JavaScript code is cached locally, this problem effectively disappears."¹⁹⁹ This would provide further motivation for a POSITA to deploy O'Reilly's teaching in the browser-enabled system of Roy and Fennell.

277. O'Reilly also discloses a variety of techniques for performing application independent data movement operations using JavaScript, including saving state with cookies and

¹⁹⁷ O'Reilly (Ex. 1006) at 166 (§10.2) (emphasis added).

¹⁹⁸ O'Reilly (Ex. 1006) at 242 (§16.1).

¹⁹⁹ O'Reilly (Ex. 1006) at 174 (§10.6.1).

sending cookies to a server²⁰⁰, loading the results of form submission into a named frame²⁰¹, caching JavaScript code²⁰², and image caching²⁰³, to name just a few examples. In fact, any example of caching necessarily involves associated data movement.

278. Thus O'Reilly discloses an application-independent model as claimed in the '719 Patent.

279. O'Reilly further teaches that it is preferable to associate logic that would interact with documents with a frame separate from the document frame:²⁰⁴

As we saw in the error handler example, JavaScript code in one window (or frame) can dynamically create an HTML document in another window (or frame). It is a lot harder for JavaScript code to dynamically create a new HTML document in its own window or frame, because doing this generally overwrites the JavaScript code itself! If your web page design calls for one static frame and two frames that have their contents dynamically updated, the static frame can contain the JavaScript code necessary to update the dynamic frames. But what if your design calls for all the frames to be dynamic? A static frame is still required, but the trick here is to create the static frame so that it is invisible! You do this by explicitly creating it at a location that is greater than 100% of the frame width or height. HTML to create such an invisible frame is shown in Example 12.5.

280. In my opinion, a POSITA would understand that an application-independent model, like a caching function, would preferably be associated with a frame separate and apart from the document frame itself, per the teachings of O'Reilly, and would deploy this strategy in an Internet-based browser enabled application.

281. Accordingly, a POSITA would find it obvious that the at least one application-independent view-generating logic and controller logic frame further has an application-

²⁰⁰ O'Reilly (Ex. 1006) at 234–41 (§15).

²⁰¹ O'Reilly (Ex. 1006) at 251–66 (§17).

²⁰² O'Reilly (Ex. 1006) at 166 (§10.2).

²⁰³ O'Reilly (Ex. 1006) at 242 (§16.1).

²⁰⁴ O'Reilly (Ex. 1006) at 207–09 (§12.4) (highlighting added).

independent model associated therewith, as O'Reilly teaches, in the Internet-based browser-enabled application Roy and Fennell teach.

282. Thus, in my opinion, Roy in combination with Fennell and O'Reilly renders obvious claims 9 and 21 of the '719 Patent.

(9) Claims 10 and 22: “wherein the at least one application-independent view-generating logic and controller logic frame serves as an application programming interface for developing views to be displayed in accordance with the application”

283. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent, and together with Fennell renders obvious claims 3 and 15, and further in combination with O'Reilly renders obvious claims 4, 8, 15, 16, and 20. Roy, in combination with Fennell and O'Reilly renders obvious dependent claims 10 and 22.

Claim 10	Claim 22
The method of claim 8, wherein the at least one application-independent view-generating logic and controller logic frame serves as an application programming interface for developing views to be displayed in accordance with the application.	The system of claim 20, wherein the at least one application-independent view-generating logic and controller logic frame serves as an application programming interface for developing views to be displayed in accordance with the application.

284. A POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell's teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

285. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O'Reilly, which provides programming techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

286. In my opinion, a POSITA would find obvious that an application-independent view-generating logic and controller logic frame serves as an application programming interface for developing views to be displayed in accordance with the application.

287. The '719 Patent describes the application-programming-interface (API) as being deployed in JavaScript as follows:²⁰⁵

In a preferred embodiment, the application programmer codes the application-specific frame model and controller logic using JavaScript, and generates the view via calls to the library API.

....

Further, the invention provides a client-side application-programming interface (API) for application developers to use when implementing dual-MVC applications. Using this API, which may be implemented in JavaScript, makes it easier to write applications.

288. Roy discloses the use of APIs to connect the application software to operating system software.²⁰⁶

289. O'Reilly likewise discloses the use of APIs to connect Java and JavaScript "through the API provided by the JSObject class."²⁰⁷ O'Reilly also discloses a number of APIs that connect JavaScript to Java applets, including LiveAudio, LiveVideo, and Live3D.²⁰⁸ Further, O'Reilly dedicates an entire chapter to LiveConnect, Netscape's name for its JavaScript-to-Java and Java-to-JavaScript interaction API.²⁰⁹

290. In my opinion, a POSITA would have found it obvious to use the application-independent view-generating logic and controller logic frame as an API for developing views to

²⁰⁵ '719 Patent (Ex. 1001) at 7:51–67 and 8:34–38 (highlighting added).

²⁰⁶ Roy (Ex. 1004) at Fig. 1 and 3:48–56 (stating that "The game software includes various application program interfaces (APIs) providing an interface to operating system software.").

²⁰⁷ O'Reilly (Ex. 1006) at 307 (§19.6.3).

²⁰⁸ O'Reilly (Ex. 1006) at 623–26 (Appendix H).

²⁰⁹ O'Reilly (Ex. 1006) at 283–312 (§19.6.3).

be displayed in accordance with the application, because the API would call the appropriate methods to generate the views needed to run the application. As Roy discloses, the APIs provided in the application software enabled interfacing with higher-level software, e.g., a runtime system. This would be required to provide a view of the model so that a user could actually interact with the application. Combined with the teachings of O'Reilly—that JavaScript APIs could connect to various software elements—it would be obvious to deploy the application-independent view-generating logic and controller logic frame as an API for developing views to be displayed in accordance with the application.

291. The frame serving as an application-independent view-generating and controller logic container is rendered obvious by O'Reilly, which describes using frames to modularize and isolate reusable application logic. By combining this teaching with the synchronization and client-local execution models of Roy and Fennell, a POSITA would predictably arrive at the subject matter claimed in claims 10 and 22. Frames inherently allow independent operation and communication with other parts of the application, making them natural candidates to contain APIs for view generation.

292. These claims are rendered obvious by combining the modular synchronization and local execution methods in Roy and Fennell with the teaching of frames as modular, reusable containers in O'Reilly. A POSITA would be motivated to configure a frame to act as an application-independent API for view generating and controller logic, as this would predictably reduce server interactions, improve modularity, and streamline application development and maintenance in client-server systems.

293. Thus, in my opinion, Roy in combination with Fennell and O'Reilly renders obvious claims 10 and 22 of the '719 Patent.

(10) Claims 11 and 23: “wherein the views are implemented in accordance with the HyperText Markup Language and the application programming interface is implemented in accordance with the JavaScript language”

294. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent, and together with Fennell renders obvious claims 3 and 15, and further in combination with O'Reilly renders obvious claims 4, 8, 10, 15, 16, 20, and 22. Roy, in combination with Fennell and O'Reilly renders obvious dependent claims 11 and 23.

Claim 11	Claim 23
The method of claim 10, wherein the views are implemented in accordance with the HyperTextMarkup Language and the application programming interface is implemented in accordance with the JavaScript language.	The system of claim 22, wherein the views are implemented in accordance with the HyperTextMarkup Language and the application programming interface is implemented in accordance with the JavaScript language.

295. A POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell's teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

296. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O'Reilly, which provides programming techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

297. In my opinion, a POSITA would find obvious that the views are implemented in accordance with HTML and the application programming interface is implemented in JavaScript.

298. First, a POSITA would understand that HTML would be used to specify the views of a browser-based application. As discussed above, O'Reilly discloses a number of view-related

features of HTML documents, and a POSITA would thus understand that the view would be implemented in HTML. For example, O'Reilly discloses:

Without a doubt, the most important feature of the Document object (and perhaps of client-side JavaScript in general) is the write() method, which allows us to dynamically generate web page content from our JavaScript programs. There are several ways that this method can be used. The most obvious is to use it within a script to output HTML into the document that is currently being parsed. This is the way it was used above to display the Document lastModified property at the bottom of the web page.²¹⁰

299. O'Reilly further explains the view-related attributes:²¹¹

The Document object has a number of properties that correspond to attributes of the <BODY> tag, and which are used to specify the colors that the document is displayed in. The bgColor property, and the BGCOLOR attribute specify the background color of the document. Similarly, the fgColor and the TEXT attribute specify the default color for text in the document. The linkColor property specifies the color of unvisited links, and vlinkColor and alinkColor specify the color of visited links and of activated links (i.e., links currently being clicked on). The LINK, VLINK, and ALINK attributes correspond to these properties.

300. O'Reilly also specifies that the "write()" method can dynamically generate an HTML document that is displayed as intended, and even specifies that it would be preferable to do so from one frame into another frame.²¹²

Without a doubt, the most important feature of the Document object (and perhaps of client-side JavaScript in general) is the write() method, which allows us to dynamically generate web page content from our JavaScript programs. There are several ways that this method can be used. The most obvious is to use it within a script to output HTML into the document that is currently being parsed. This is the way it was used above to display the Document lastModified property at the bottom of the web page. Be aware that you can only output HTML to the current document while that document is being parsed. That is, you can only call document.write() from within <SCRIPT> tags, because these scripts are executed as part of the document parsing process. In particular, if you call document.write() from an event handler, you will end up overwriting the current document (including its event handlers), instead of appending text to it. Although you can't usefully write to the current document from an event handler, there is no reason you can't write to a document in another window or frame, and doing so can be a very useful technique for multiwindow or multiframe web sites.

²¹⁰ O'Reilly (Ex. 1006) at 224 (§14.1.2).

²¹¹ O'Reilly (Ex. 1006) at 222–23 (§14.1.1) (highlighting added).

²¹² O'Reilly (Ex. 1006) at 224 (§14.1.2) (highlighting added).

301. Second, a POSITA would understand that the API would be implemented in JavaScript. For example, O'Reilly discloses the use of APIs to connect various software functionalities, for example, that Java can interact with JavaScript "through the API provided by the JSObject class."²¹³ O'Reilly also discloses a number of APIs that connect JavaScript to Java applets, including LiveConnect, LiveAudio, LiveVideo, and Live3D.²¹⁴

302. Further, a POSITA would understand that JavaScript and HTML work together. For example, O'Reilly explains that both JavaScript and HTML would be used to create framed layouts, stating: "JavaScript does not define methods that allow you to directly create and manipulate frames within a browser window. However, the ability to dynamically generate HTML allows you to programmatically write the HTML tags that will create any desired frame layout."²¹⁵

303. Accordingly, a POSITA would find it obvious that the views of the '719 Patent would be implemented in accordance with the HyperTextMarkup Language and the application programming interface would be implemented in JavaScript, as O'Reilly teaches, in the Internet-based browser-enabled application Roy and Fennell teach.

304. Thus, in my opinion, Roy in combination with Fennell and O'Reilly renders obvious claims 11 and 23 of the '719 Patent.

²¹³ O'Reilly (Ex. 1006) at 307 (§19.6.3).

²¹⁴ O'Reilly (Ex. 1006) at 623–26 (Appendix H).

²¹⁵ O'Reilly (Ex. 1006) at 19 (§1.3.2).

(11) Claims 12 and 24: “wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand”

305. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent. Roy, in combination with Fennell and O'Reilly, discloses or renders obvious dependent claims 12 and 24.

Claim 12	Claim 24
The method of claim 1, wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.	The system of claim 13, wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.

306. In addition, the requirement that downloads be “upon demand” does nothing to add novelty to these claims.

307. For example, web pages employing HTML and JavaScript that contain a model, view-generating logic and controller logic were routinely downloaded “upon demand” from a web server to a web browser by at least 1996. By the time the '719 Patent was filed, a POSITA would have extensive knowledge and experience downloading a model, as well as the view-generating and the controller logic associated with that model, since that capability had been present in commercial web browsers (e.g., Netscape Navigator 3.0, released in 1996) for years by that point.

308. A POSITA would have been motivated to modify the Internet-based application Roy describes based on Fennell’s teachings about using a browser-enabled device for the reasons discussed above, the discussion of which is incorporated herein.

309. A POSITA would have likewise been motivated to combine an Internet-based browser-enabled application with the teachings of O'Reilly, which provides programming

techniques and examples for using JavaScript (including how to interact with Java applets) in browsers for the reasons discussed above, the discussion of which is incorporated herein.

310. In my opinion, a POSITA would have found it obvious that the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.

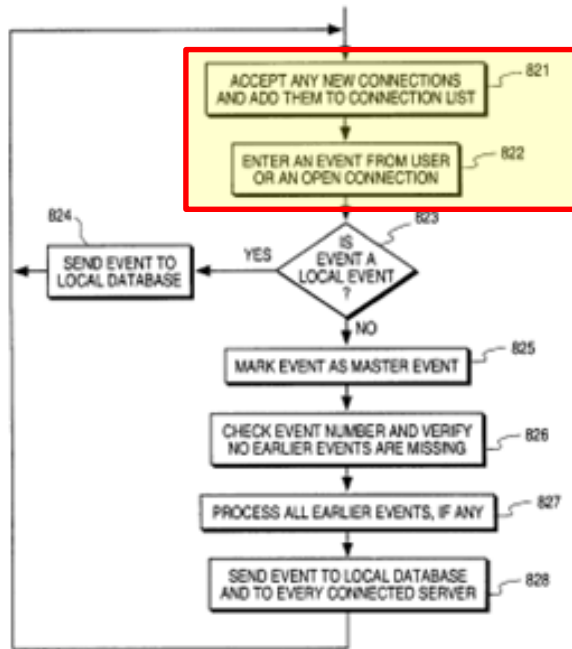
311. To start, Roy discloses that client computers may connect to a server to join the session, and the route to the master for that client would not be created until and unless there was a need, i.e., to join the game. For example, Roy discloses “a technique for establishing a connection to a computer system having multi-player game software” as depicted in Figure 7:²¹⁶



312. Similarly, Figure 8B illustrates a routine used by a master event server for opening connections, which is not enabled until there is a request for new connections.²¹⁷

²¹⁶ Roy (Ex. 1004) at Figure 7 and 2:41–43.

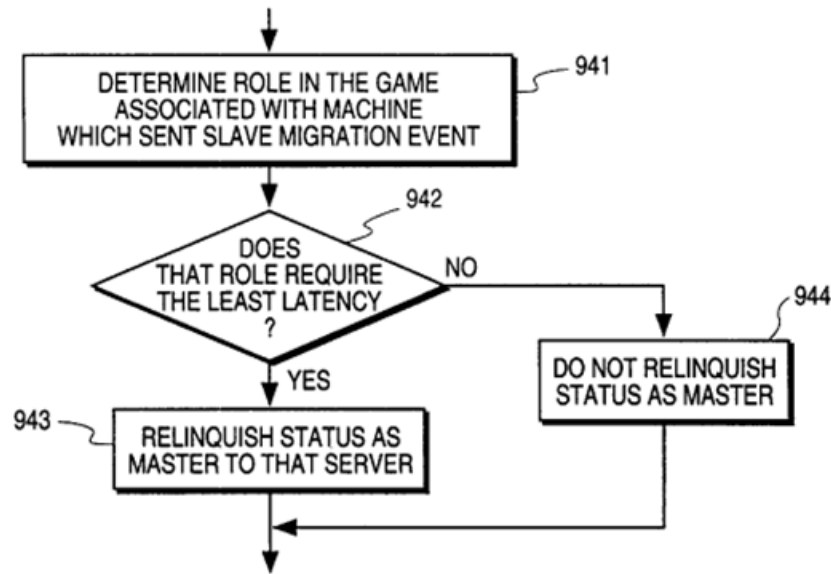
²¹⁷ Roy (Ex. 1004) at Figure 8B (highlighting added) and 7:1–2.



313. Additionally, Roy discloses that server data (including the model) can be migrated to any client computer on demand per a request:

Regardless of which device has the actual authority to approve of migration of the master event server, the migration process can be initiated by a request or command transmitted over the network by a slave event server. The request or command is a special event known as a ‘slave migration event.’ The master event server initiates the migration process by sending a ‘master migration event’. When a slave event server receives a slave migration event from another slave event server, it adds its identifier to the end of the event message. Hence, the slave migration event gets annotated with the complete path it follows to reach the master event server. The master migration event sent in reply also contains the same path information. Accordingly, interceding slave event servers can correctly route the master migration event to the slave event server that transmitted the slave migration event and update their respective RTM information.²¹⁸

²¹⁸ Roy (Ex. 1004) at 7:43–59 (highlighting added).



219

FIG. 9C illustrates a routine by which a master event server can determine whether to relinquish its status as master event server. Upon receiving a slave migration event, the master event server identifies the role in the game that is associated with the slave event server that sent the slave migration event (step 941). If that role requires the lowest latency of any role in the game, then it is appropriate to migrate the master event server. Accordingly, the master event server relinquishes its status as master event server (step 943). Otherwise, the master event server retains its status as master (step 944).²²⁰

314. In light of the foregoing, it would be understood by or at least rendered obvious to a POSITA that, when a new user joined the session, a slave copy of the database would be transferred to the new user's computer (i.e., at least a subset of the model would be downloaded from the server to the client upon demand).²²¹ Roy also explains that the event server of a new user can "connect[] to another event server that is already part of the game."²²²

²¹⁹ Roy (Ex. 1004) at Fig. 9C.

²²⁰ Roy (Ex. 1004) at Fig. 9C and 8:41–51 (highlighting added).

²²¹ Roy (Ex. 1004) at 4:51–55.

²²² Roy (Ex. 1004) at 5:50–64.

315. Roy explains that one of the benefits of its distributed system was “reducing bandwidth consumption on the network,” which a POSITA would understand would also be accomplished by only downloading data “on demand.”

“The display 74 of computer system 62 is updated by rendering commands 73 from the slave copy 72 of master database 71, rather than from the master database 71 itself, thereby reducing bandwidth consumption on the network.”²²³

316. O’Reilly teaches that downloading files takes time and can cause delays, and accordingly, a POSITA would know to only download the data if it was actually needed (or in demand): “Downloading an external file of JavaScript code, even over a relatively fast modem connection, can cause noticeable delays in the loading and execution of a web page. Of course, once the JavaScript code is cached locally, this problem effectively disappears.”²²⁴

317. Fennell expressly teaches that online applications could be deployed over a browser with “on demand” software. Specifically, Fennell teaches that in its networked system, a “‘terminal’ refers to any suitable input/output device such as . . . a smart terminal such as a World Wide Web browser with software ‘applets’ (written in, e.g., Sun Microsystems’ Java language) downloaded on demand[.]”²²⁵

318. Accordingly, a POSITA would find it obvious that the “at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.”

319. Thus, in my opinion, Roy in combination with Fennell and O’Reilly renders obvious claims 12 and 24 of the ’719 Patent.

²²³ Roy (Ex. 1004) at Fig. 5A and 5:17–49 (highlighting added).

²²⁴ O’Reilly (Ex. 1006) at 174 (§10.6.1).

²²⁵ Fennell (Ex. 1005) at 3:3–15 (emphasis added).

D. Roy, in Combination with Fennell, Renders Obvious Claims 3 and 15 of the '719 Patent

320. As discussed above, Roy anticipates or renders obvious each of the independent claims of the '719 Patent. Roy, in combination with Fennell, renders obvious dependent claims 3 and 15.

Claim 3	Claim 15
The method of claim 1, wherein the client performs the one or more portions of the application in accordance with browser software running thereon.	The system of claim 13, wherein the client processor performs the one or more portions of the application in accordance with browser software running thereon.

321. To start, the '719 Patent is agnostic to the type of “browser software” that it claims would work with its system: “The system comprises a client computer system which is configured to run web browser software 1000. It is to be appreciated that any suitable web browser software may be employed and that the methodologies of the invention are not limited to any particular web browser software.”²²⁶ Accordingly, any browser software disclosed would render obvious these claims.

322. As discussed above, Fennell discloses that an Internet-based application could be deployed over a browser.²²⁷ A POSITA would have been motivated to combine Roy and Fennell, as discussed above. Accordingly, the combination of Roy and Fennell renders obvious that the application could be operated such that the client performs the one or more portions of the application in accordance with browser software running thereon, rendering obvious claims 3 and 15 of the '719 Patent.

²²⁶ '719 Patent (Ex. 1001) at 4:51-55.

²²⁷ Fennell (Ex. 1005) at 3:11–19 (“A terminal 110 may be . . . a smart terminal such as a World Wide Web browser with software ‘applets’ (written in, e.g., Sun Microsystems’ Java language) downloaded on demand.”).

IX. Secondary Considerations

323. I understand that a patent owner may offer objective evidence of nonobviousness, also called secondary considerations, that the asserted claims of its patent are not obvious. I understand this evidence may include long-felt but unmet need, independent invention and failure of others, and commercial success. I also understand that a patent owner has to prove a connection between the secondary considerations and the patent claims in order to show nonobviousness.

324. I am not aware of evidence of secondary considerations of nonobviousness showing the required connection that would change my opinion that the '719 Patent claims are invalid for the reasons stated in this declaration.

325. The '719 Patent claims its solution is superior to a number of prior art systems including “dumb terminals,” “fat-client applications,” “applets,” ““thin-client’ application architecture,” “DHTML,” “HTML (HyperText Markup Language) which includes JavaScript and Cascading Style Sheets.”²²⁸

326. However, the claimed improvements over these technologies already existed at the time the '719 Patent was filed, for example, as discussed above, Roy discloses a “dual-MVC” system that reduces round trips to the server as claimed in the independent claims of the '719 Patent. §§ VII.A; VIII.A. Further, the '719 Patent’s asserted benefit of not having “application installation or persistence on the client” is not actually claimed in the '719 Patent.²²⁹ And as discussed above, Roy, Fennell, and O’Reilly together disclose all the subject matter that *is* claimed, including HTTP (e.g., Claims 2 and 14), browser software (Claims 3 and 15), frames (e.g., Claims 4–6, 8–10, 16–18, 20–22), invisible frames (e.g., Claims 7 and 19), HTML and JavaScript (e.g.,

²²⁸ '719 Patent (Ex. 1001) at 1:9–2:67.

²²⁹ '719 Patent (Ex. 1001) at claims 1–26 (neither “application installation” nor “persistence on the client” present).

Claims 11 and 23), API controls (e.g., Claims 10 and 22), downloading on demand (e.g., Claims 12 and 24), and application independent features including caching, view-generating logic, and controller logic (e.g., Claims 8–10, 20–22). §§ VII.C–D.

327. In contrast to this evidence that others successfully practiced what is claimed in the '719 Patent, I am not aware of evidence showing failure of others to practice the claims of the '719 Patent, especially as multiple prior art references disclose the successful implementation of the subject matter claimed.

328. I understand that IBM asserted the '719 Patent against third-party gaming company Zynga Inc. I understand that Zynga's witness Dr. Douglas Kidder testified that IBM had not signed a single-patent license to the '719 Patent alone, but only included the '719 Patent in portfolio licenses.²³⁰ To my knowledge, this testimony was unrebutted. I understand that portfolio licenses covering many patents and/or patent applications are considered to be of limited value in determining nonobviousness.

329. If IBM offers evidence or argument regarding secondary considerations, including in any Patent Owner Response or expert declaration, I reserve the right to respond by supplementing, amending, or modifying my opinions as stated herein.

X. Conclusion

330. It is my opinion that all the claims of the '719 Patent are invalid. Specifically, it is my opinion that claims 1, 13, 25, and 26 are anticipated or rendered obvious by Roy. Further, it is my opinion that claims 2, 4–12, 14, and 16–24 are rendered obvious by Roy in combination with

²³⁰ *International Business Machines Corp. v. Zynga Inc.*, Case No. 22-cv-590-GBW, Sept. 12, 2024 Trial Tr. (D. Del.) (Trial Tr. at 126:22–127:1).

Fennell and O'Reilly. Finally, it is further my opinion that claims 3 and 15 are rendered obvious by Roy in combination with Fennell.

331. This declaration reflects my opinions as currently held. However, my analysis may continue, and I may consider additional information that may result in additional observations.

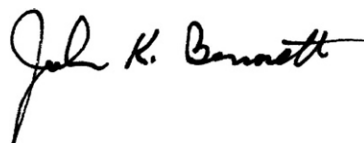
332. If asked, or to more fully explain my opinions expressed in this declaration, I reserve the right to rely on various portions of the references that I have discussed in this declaration that I have not already explicitly cited in this declaration.

333. In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be asked to provide testimony related to my opinions, and will do so, if required.

I hereby declare that all statements made herein are to the best of my knowledge true and correct. Further, I understand that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Date: January 30, 2025

By:

A handwritten signature in black ink that reads "John K. Bennett". The signature is written in a cursive style with a large, stylized "J" and "B".

John K. Bennett, Ph.D., P.E.

John K. Bennett

Education

- **University of Washington**
Ph.D., Computer Science
M.S., Computer Science
- **Rice University**
MEE, Electrical and Computer Engineering
BSEE, Electrical and Computer Engineering

Professional Engineering Registration: State of Texas, P.E. No. 75525

Memberships

ACM, SIGOPS, SIGARCH, SIGMETRICS, SIGCOMM, IEEE (Senior Life Member), IEEE Computer Society.

Research Interests

Human-Centered Design and Innovation; Information and Communication Technology for Development; Computer Science and Engineering Education; Distributed Computing; Distributed Robotic Sensors; Parallel Shared Memory Computing.

Experience

University of Colorado Denver Anschutz Medical Campus	Associate Vice Chancellor for Innovation Initiatives, 2014-2020 Inworks Director (see www.inworks.org), 2014-2020 Professor of Computer Science & Engineering, 2014-2020
University of Colorado Boulder	Archuleta Professor of Computer Science (retired), 2000-2020 Professor of Electrical, Computer & Energy Engineering, 2000-2020 Director of the ATLAS Institute, 2007-2013 Associate Dean of Engineering for Education, 2002-2007
Rice University	Master of Richardson College, 1996-2000 Associate Professor of Electrical and Computer Engineering, 1994-2000 Assistant Professor of Electrical and Computer Engineering, 1988-1994
University of Washington	Visiting Associate Professor of Computer Science, 1996 Predoctoral Lecturer in Computer Science, Winter, 1983 Research Assistant in Computer Science, 1980-1982, 1987 Research Assistant in High Energy Physics, 1979-1980
Pacific Mtn. Research, Inc.	President, 1982-1987
US Navy	Sr. Ship Superintendent, Puget Sound Naval Shipyard, 1977-1979 13th Naval District Diving and Salvage Officer, 1978-1979 Electrical Officer & DCA, USS Paul F. Foster (DD-964) 1974-1977

Theses Supervised

1. Katherine Goodman. The Transformative Experience in Engineering Education. (Ph.D. Dissertation. December, 2015). (jointly supervised with Jean Hertzberg)
2. Kara Behnke. Gamification in Introductory Computer Science. (Ph.D. Dissertation. December, 2015).
3. Leslie Dodson. A Foggy Desert: Equitable Information Flow for a Fogwater System in Southwest Morocco. Ph.D. Dissertation. May, 2014. (jointly supervised with S. Revi Sterling)
4. Heather Underwood. PartoPen: Improving Maternal Labor Monitoring with Digital Pen Technology. Ph.D. Dissertation. December, 2013. (jointly supervised with S. Revi Sterling)
5. Edwige Simon. The Impact of Online Teaching on Higher Education Faculty's Professional Identity and the Role of Technology: The Coming of Age of the Virtual Teacher. Ph.D. Dissertation. December, 2012.
6. S. Revi Sterling. Advancement Through Interactive Radio. Ph.D. Dissertation. December, 2008.
7. Joseph Dunn. Secure Sharing Across Trust Boundaries. Ph.D. Dissertation. August, 2007.
8. Brian Shucker. Control of Distributed Robotic Macrosensors. Ph.D. Dissertation. July, 2006.
9. L. Daniel Crawl. Affinity Directed Mobility. Ph.D. Dissertation. July, 2006.
10. Carlos Matos. Adapting a PDA to Enhance Voice Treatment in Parkinson Disease. MS thesis. December, 2001.
11. Hazim Abdel-Shafi. Reliable Parallel Computing. Ph.D. Dissertation. May, 2000.
12. Damian Dobric. Efficient User-Level Communication. MS thesis. May, 2000.
13. Vasilios Balabanos. Optimization of Logic Synthesis. MS thesis. May, 2000.
14. W. Evan Speight. Efficient Runtime Support for Cluster-Based Distributed Shared Memory Multiprocessors. Ph.D. Dissertation. July, 1997.
15. Yanyang Xiao. Memory Architecture in Multi-channel Optically Interconnected Distributed Shared Memory Multiprocessor Systems. MS thesis. July, 1996.
16. Dierdre Hamilton. Effectiveness and Performance Analysis of a Class of Parallel Robot Controllers with Fault Tolerance. Ph.D. Dissertation. May, 1996. (jointly supervised with Ian Walker)
17. Rajat Mukherjee. The Interaction of Architecture and Operating System in the Design of a Scalable Shared Memory Multiprocessor. Ph.D. Dissertation. October, 1994.
18. Kenneth Mark Maxham. ASPEN: High-Performance Hardware Support for Distributed Shared-Memory. MS thesis. May, 1994.
19. W. Evan Speight. ParaView: Performance Debugging Through Visualization of Shared Data. MS thesis. August, 1993.
20. John Carter. Munin: Efficient Distributed Shared Memory Based On Multi-Protocol Release Consistency. Ph.D. Dissertation. December, 1992. (jointly supervised with Willy Zwaenepoel)
21. Dierdre Hamilton. Performance and Reliability of a Parallel Robot Controller. MS thesis. May, 1992. (jointly supervised with Ian Walker)
22. Jay Greenwood. The Design of a Scalable, Hierarchical-Bus, Shared-Memory Multiprocessor. MS thesis. May, 1992.
23. Rajat Mukherjee. Simulation of Shared Memory Parallel Systems. MS thesis. May, 1990.

Patents

1. Apparatus for Electrical Connection of Glove Monitor to Patient, U.S. Patent No. 5,658,277, August 19, 1997.
2. Enhanced Monitoring Device for Surgical Gloves and Other Barriers, U.S. Patent No. 5,389,097, February 14, 1995.

Honors and Awards

- Elected Professor Emeritus, University of Colorado (2021).
- American Speech-Language-Hearing Association (ASHA) Editor's Award (with co-authors) for the paper "Innovative technology for the assisted delivery of intensive voice treatment (LSVT®LOUD) for Parkinson disease" in *American Journal of Speech-Language Pathology*, 21(4):354-67, November, 2012.
- PhD Student Heather Underwood received First Place in ACM 2013 Student Research Competition Grand Finals (Graduate Category).
- University of Washington CSE Alumni Achievement Award (2012)
- Archuleta Endowed Professorship (2007)
- Elected to membership in Eta Kappa Nu (2007)
- Fellow of the ATLAS Institute (2005)
- Finalist for San Jose Tech Museum of Innovation Award (for work related to PDA-assisted treatment of Parkinson Disease) (2002)
- W.M. Keck Foundation National Award for Engineering Teaching Excellence (1996)
- George R. Brown Award for Innovative Teaching (1995)

Research Funding

- PartoPen: A Tool for Improving Maternal Outcomes. Bill and Melinda Gates Foundation. (\$100,000), 2012-2014 (in support of the doctoral research of Heather Underwood).
- Telehealth Colorado: Rural and Remote Access to Speech and Language Therapy for Toddlers, Preschoolers, and Older Students with Cochlear Implants. (\$4800). 2010-2011.
- NSF-CRA ICTD sub award # CIF-234 130-94286 # 1546246 (\$140,000), 2009-2011.
- CableLabs Doctoral Fellowship. CableLabs. (\$175,000), 2009-2013.
- AIR – Advancement through Interactive Radio. Microsoft Corporation Digital Inclusion Program (\$150,000), 2006-2008.
- Automatic Distributed Unmanned Vehicles Coordination and Control. Raytheon Corporation (\$25,000), 2005-2006.
- Rehabilitation Engineering Research Center On Recreational Technologies And Exercise Physiology: Project Rec-Tech. NSF (\$4,500,000), 2002-2007, with J. Rimmer et al.
- Collaborative Research: Affinity Directed Mobility for Location Independent Data Access. NSF (\$346,995), 2002-2005.
- PDA-Enhanced Speech Treatment for Parkinson Disease. NIH (\$302,701), 2001-2003, with Lorraine Ramig.
- Using a Portable Computing Device to Assist Persons with Parkinson Disease. Coleman Institute (\$25,000), 2001-2002, with Lorraine Ramig.
- The Bifrost Location Independent Computing System. Microsoft Corporation (\$185,000), 2000-2001, with Evan Speight, Cornell University.
- Optimizing VHDL Intermediate Form. DARPA (\$751,000), 1997-2001, with Keith Cooper and Linda Torczon.
- Efficient Shared Memory Parallel Computing on Personal Computer Clusters. Compaq Computer Corporation, (\$260,000), 1998-2000.
- Efficient Shared Memory Parallel Computing on Personal Computer Clusters. Texas Advanced Research Program / ATP (\$203,749), 1998-2000.
- Support for the Brazos Project. Intel Corporation (\$14,060), 1999.
- Support for the Brazos Distributed Shared Memory Project. Microsoft Corporation (\$50,000), 1998-1999.
- Support for the Brazos Project. Intel Corporation (\$38,880), 1998.
- Support for the Brazos Distributed Shared Memory Project. Tandem Corporation (\$30,000), 1998.

- Distributed Shared Memory Research. Microsoft Corporation (\$18,000), 1997-1998.
- A Medium Scale, Tightly Coupled, Shared-Memory Multiprocessor. NSF CISE Research Instrumentation (\$82,932), with Sarita Adve et al., 1996.
- Multiprocessor Cluster Computing. NSF (\$1,050,000), with Willy Zwaenepoel et al., 1995-2000.
- Physical Properties of Latex Products. Ansell Medical (\$50,000), 1995-1996
- Performance Debugging of Shared Memory Parallel Programs, with Sarita Adve. Texas Advanced Research Program / ATP (\$207,514), 1994-1996.
- Efficient Parallel Computing on Networks of Personal Computers, with Willy Zwaenepoel. Texas Advanced Research Program / ATP (\$480,000), 1994-1996.
- Physical Properties of Latex Surgical Gloves. Regent Hospital Products, Ltd. (\$42,500), 1994-1995
- Management of Mobility-Related Data Using Shared-Memory Techniques. Bell Northern Research (\$90,000), 1992-1995.
- Physical Properties of Latex Surgical Gloves. Smith and Nephew Perry (\$50,000), 1993-1995
- A Software Environment for a Scalable Shared Memory Processor. NASA AMES (\$66,000), 1991-1994.
- Parallel Fault-Tolerant Robot Control. NASA JSC (\$66,000), 1992-1995.
- The Interaction of Architecture and Operating System in the Design of a Scalable Shared Memory Multiprocessor. IBM (\$38,500), 1991-1993.
- Texas Geophysical Parallel Computation Project. State of Texas, PI: Ken Kennedy. (\$100,000), 1990-1992.
- Research Experience for Undergraduates. NSF (\$4,400), 1991-1993.
- Adaptive Software Cache Management for Distributed Shared Memory Architectures. NSF Research Initiation Award (\$60,000), 1990-1993.
- Special Graduate Student Education and Research Award. (NSF, \$6,000), 1990-1992.

Education Funding

- Support for the ATLAS Teaching Laboratories. Estate of William J. Pryde. (\$350,000), (2011).
- Support for ATLAS Digital CurrenTs Summer Outreach. Musk Foundation. (\$10,000), (2010).
- Support for the CU Western Colorado Rural Engineering Program. Daniels Fund (\$15,000), 2007; Shell Corporation (\$15,000), 2007; GE Foundation (\$100,000), 2006; Lockheed Martin Corporation (\$15,000), 2006; Microsoft Corporation (\$20,000), 2005; Bechtel Foundation (\$150,000), 2005-2009.
- Rummel Creek Elementary University: A K-5 Institution of Higher Learning, with Shirley Lincoln. R.J.R. Nabisco Next Century School Fund (\$750,000), 1992-1995.
- Viewlogic Systems Corporate Grant. (\$846,000), 1990-1995.
- Enhancement of the Rice Computer Systems and Digital Design Laboratories. Hewlett-Packard Company Corporate Grant (\$62,205), 1989.

Professional Activities

- Volunteer & Professor of the Practice, Colorado School of Mines (2022-)
- Member of the Board of Directors, MindSpark Learning (2016-2021)
- Program Committee 2013 IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD2013)
- Program Committee 3rd Annual Symposium on Computing for Development (DEV 2013)
- Program Committee 2011 IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD2011)
- Microsoft Imagine Cup National Judge (2011)

- Program Committee Regional Chair (North America), 2009 IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD2009)
- President of the Governing Board, Engineers Without Borders-USA (2006-2009)
- Member of the Board of Directors, National Center for Women and IT (2007-2018)
- Member of the Academic Alliance, National Center for Women and IT (2004-)
- Member of the Governing Board, Engineers Without Borders-USA (2003-2009)
- Program Committee Member, 2007 IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD2007)
- Program Committee Member, 2004 Assets: The Sixth International ACM SIGCAPH Conference on Assistive Technologies
- Program Committee Member, 2002 Assets: The Fifth International ACM SIGCAPH Conference on Assistive Technologies
- Program Committee Member, 2000 International Conference on High Performance Parallel Computing
- Program Committee Member, 1998 International Conference on Parallel Processing
- Program Committee Member, 1998 Windows NT Research Symposium
- Program Committee Member, 1996 Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)
- Program Committee Member, 1996 International Conference on Distributed Computing Systems
- General Chair, 1995 Symposium on Operating Systems Principles
- Reviewer for:
ACM Transactions on Computer Systems, IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Robotics, Journal of Parallel and Distributed Computing, Software Practice and Experience, IEEE Computer Magazine, Conference on Object Oriented Programming, Systems, Languages, and Applications, International Symposium on Computer Architecture, Symposium on High Performance Computer Architecture, Symposium on Operating Systems Principles, International Conference on Architectural Support for Programming Languages and Operating Systems, Hawaii International Conference on System Sciences, Symposium on the Frontiers of Massively Parallel Computation, International Conference on Information and Communication Technologies and Development, NSF, and NSERC.

Courses Taught

At the University of Colorado: (*avg. student instructor evaluation over all courses last 5 years: 5.6/6.0*)

- IWKS 4800/5800 – Innovation and Entrepreneurship
- IWKS 3400/5400 – Game Development
- IWKS 3300/5300; CSCI 3800 – NAND to Tetris: Building a Modern Computer from First Principles
- CSCI/ATLS 1220; IWKS 2300/5350 – Virtual Worlds: Introduction to Computer Science
- GEEN 1400 - Introduction to Engineering Design
- GEEN 1500 - Introduction to Engineering
- ATLS 2000 – The Meaning of Information Technology
- CSCI 2830 – The Profession of Computer Science
- ATLS 4519; Computer Game Development
- ATLS 5519 - Advanced Computer Game Development
- CSCI 5573 - Advanced Operating Systems
- ECEN 4633/5633 – Hybrid Embedded Systems
- ATLS 7000 - ATLAS PhD Seminar

- CSCI 7000 - Advanced Topics: Mobile Computing & Interdisciplinary Graduate Seminar
- Faculty supervisor for 85 undergraduate independent study projects

At Rice University:

- ELEC 201 - Introduction to Engineering Design
- ELEC 424 - Computer System Design
- ELEC/COMP 425 - Computer Systems Architecture
- ELEC 426 - Digital System Design
- ELEC/COMP 525 - Advanced Computer Architecture
- ELEC 625 - High Performance Microprocessor Design
- ELEC 693 - Advanced Topics: Cluster Multiprocessors
- ELEC 693 - Advanced Topics: Operating System/Architecture Interaction
- ELEC 693 - Advanced Topics: Shared Memory Multiprocessing
- ELEC 693 - Advanced Topics: Techniques for Tolerating Memory Latency
- ELEC 694 - Advanced Topics: Issues in Parallel Computing
- ELEC 694 - Advanced Topics: Shared Memory Parallel Programming
- Faculty supervisor for 82 undergraduate independent study projects

University Activities and Service

At the University of Colorado Denver | Anschutz Medical Campus:

- Chancellor's Distinguished Lecture Committee (2015-2020)
- External PUEC Member, College of Architecture and Planning (2015-2020)
- CU System Academic Online Committee (2015-2020)
- Inworks Director (2014-2020)
- OSR Creative Research Collaborative Steering Committee (2014-2020)
- Chancellor's Online "Strike Force" (2014-2015)
- Member of the External Advisory Council, National Center for Women and IT (NCWIT) (2016-2019)

At the University of Colorado Boulder:

- Faculty Advisor to the CU Engineering Fellows (2005-2020)
- Post Tenure Review Committee, Dept. of Computer Science (2014)
- College of Music Dean's External Evaluation Committee (2012-13)
- College of Architecture and Planning Joint Campus Review Committee (2011-2012)
- Academic Review and Planning Advisory Committee (2008-2012)
- Law School Dean's External Evaluation Committee (2006-2012)
- Faculty Coordinator, CU Engineering Degree at Colorado Mesa University (2005-2010)
- Silicon Flatirons Advisory Board Member (2008-2014)
- Interdisciplinary Telecommunications Program Advisory Board (2007-2014)
- Computer Science Primary Unit Evaluation Committee (2007-2014)
- Co-Chair, College of Information Task Force (2009-2010)
- Flagship 2030 Facilities Task Force (2008)
- Provost's Assessment Committee (2004-2006), Co-chair (2005)
- Chair, CEAS Undergraduate Education Council (2002-2007)
- Chair, CEAS Graduate Education Council (2002-2007)
- Engineering Dean Search Committee (2000-2002)
- Academic Affairs Budget Advisory Committee (2001-2002)
- Faculty Teaching Excellence Program New Faculty Mentor (2001-2002)

- Faculty Teaching Excellence Program Summer Institute Steering Committee (2001-2002)
- Chair, Computer Science Search Committee (2005-2006)
- Chair, Computer Science Affiliates Committee (2000-2002)
- CS Department Executive Committee (2000-2002, 2004-2006, 2007)
- CS Department Undergraduate Curriculum Committee (2000-2002)
- CS Department Resource Committee (2000-2002)
- CS Department RI Grant Steering Committee (2000-2002)
- National Judge, FIRST Robotics Competition (2006)
- Regional Judge Advisor, FIRST Robotics Competition (2005-2006)
- Volunteer Judge, FIRST Robotics Competition (2004)
- Habitat for Humanity volunteer

At Rice University:

- University Alcoholic Beverage Policy Advisory Committee (1996-2000), Chair (1997-2000)
- New College Construction User Liaison (1998-2000)
- Chair, New College Planning Committee (1998-2000)
- Chair, New College Populating Committee (1998-2000)
- Chair, Ad Hoc Thresher Advisory Committee (1997-1998)
- Jones School of Management Strategic Planning Steering Committee (1996-1997)
- University Undergraduate Curriculum Committee (1996-1997)
- University Crisis Management Team (1996-1998)
- Faculty Sponsor, Women's Lacrosse (1997-2000)
- Faculty Sponsor, Latter Day Saints Student Association (1997-2000)
- Faculty Sponsor, Women's Rugby (1995-2000)
- Faculty Sponsor, Men's Rugby (1998-2000)
- Resident Faculty Associate, Wiess College (Spring 1988)
- Faculty Associate, Wiess College (1988-2000) (Outstanding Associate in 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996)
- Engineering Divisional Advisor, Wiess College (1989-1996)
- Computer Systems Engineering Undergraduate Advisor (1989-1990)
- Computer Systems Engineering Graduate Advisor (1990-1995)
- Jones School of Management Friedkin Chair Search Committee (1994-1996)
- University Computer Committee (1988-1993)
- University Committee on Community Affairs (1993-1994)
- ECE Department Colloquia Chairman (1988-1990)
- ECE Department Undergraduate Committee (1989-1990)
- ECE Department Graduate Committee (1990-1995, 1996-1997), Chairman (1991-1992, 1994-1995)
- ECE Department Comprehensive Qualifying Exam Committee (1990-1995, 1996), Chairman (1991-1992, 1994-1995)
- ECE Department Computer Engineering Faculty Recruiting Chairman (1991-1994)
- ECE Department Computer Committee (1992-1995)
- ECE Department Corporate Affiliates Committee (1990-1995), Chairman (1994-1996)
- Engineering Facilities Committee (1988-1995)
- MBA / Professional Masters Joint Degree Planning Committee (1989)

Publications

1. Kelsey L. Ford, Jenn Leiferman, Bruno Sobral, John K. Bennett, Susan L. Moore, Sheana Bull. "It depends:" a qualitative study on digital health academic-industry collaboration. In mHealth 2021; 7:57 (20 October 2021).

2. Kara A. Behnke, Brittany Ann Kos and John K. Bennett. Computer Science Principles: Impacting Student Motivation & Learning Within and Beyond the Classroom. In *Proceedings of the Twelfth annual ACM International Computing Education Research (ICER 2016)*, Melbourne, VIC, Australia. September 2016
3. Katherine Goodman, Heather Underwood and John K. Bennett. Inworks: Making Things that Matter. In *Proceedings of the 2016 ASEE Annual Conference (ASEE 2016)*. New Orleans, Louisiana. June 2016.
4. Brittany Ann Kos, Kara C. Behnke, John K. Bennett, and Diane Sieber. Computer Science Principles: An Introductory Computer Science Course for Non-Majors. In *Proceedings of the 20th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2015)*. Vilnius, Lithuania, July 2015.
5. Heather Underwood, John Ong'ech, Maya Appley, Sara Rosenblum, Addie Crawley, S. Revi Sterling and John K. Bennett. Using Digital Pens for Maternal Labor Monitoring: Evaluating the PartoPen in Kenya. In *Communications in Computer and Information Science*, Springer-Verlag, 2015.
6. Kara Behnke, Meg Leta Ambrose and John K. Bennett. Must be 13 to Play: Addressing Children Participation in Networked Games. In *Proceedings of the 9th International Conference on the Foundations of Digital Games*, April, 2014.
7. Katherine Goodman and John K. Bennett. Modeling the Serial Position Effect: Using the emergent Neural Network Simulation System. In *Proceedings of the 5th International Conference on Bioinformatics Models, Methods and Algorithms*, Angers, France, March, 2014.
8. Heather Underwood, John Ong'ech, Maya Appley, Sara Rosenblum, Addie Crawley, S. Revi Sterling and John K. Bennett. PartoPens at the Point of Care: Evaluating Digital Pen-Based Maternal Labor Monitoring in Kenya. In *Proceedings of the 7th International Conference on Health Informatics*, Angers, France, March, 2014. Awarded Best Paper, Student First Author.
9. Leslie Dodson, S. Revi Sterling and John K. Bennett. Minding the gaps: Cultural, technical and gender-based barriers to mobile use in oral-language Berber communities in Morocco. In *Proceedings of the Sixth IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD2013)*.
10. Heather Underwood S. Revi Sterling and John K. Bennett. The PartoPen in Practice: Evaluating the Impact of Digital Pen Technology on Maternal Health in Kenya. In *Proceedings of the Sixth IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD2013)*.
11. S. Revi Sterling and John K. Bennett. "The Case for Interdisciplinarity: Lessons from the Field." In *Proceedings of the IEEE Global Humanitarian Technical Conference (GHTC 2013)*. San Jose, USA. October 2013.
12. Leslie Dodson, S. Revi Sterling and John K. Bennett. The Mobile Utility Gap: SMS use by Low-Literate Berber Women in Morocco. In *Proceedings of the Third Workshop on Intelligent User Interfaces for Developing Regions: Users, Problems and Technologies*, Santa Monica, California, March 2013.
13. Leslie Dodson, S. Revi Sterling and John K. Bennett. Considering failure: Eight years of ITID research. In *Information Technology for International Development*, 9(2), 2013, 19-34.
14. Heather Underwood S. Revi Sterling and John K. Bennett. The PartoPen in training and clinical use: Two preliminary studies in Kenya. In *Proceedings of the 6th International Conference on Health Informatics* (part of the International Joint Conference on Biomedical Engineering Systems and Technologies), Barcelona, Spain, February, 2013. Awarded Best Paper, Student First Author.
15. Leslie Dodson, S. Revi Sterling and John K. Bennett. Low-literacy and texting: Rural women in Morocco. In *Proceedings of the 2013 UNESCO Symposium on Mobile Learning*. Paris, France, February 2013.

16. Heather Underwood S. Revi Sterling and John K. Bennett. The design and implementation of the PartoPen maternal health monitoring system. In *Proceedings of the 3rd ACM Symposium on Computing for Development (DEV 2013)*, Bangalore, India, January 2013.
17. E. Halpern, L. O. Ramig, C. E. C. Matos, J. A. Petska-Cable, J. L. Spielman, J. M. Pogoda, P. M. Gilley, S. Sapir, J. K. Bennett, and D. H. McFarland. Innovative technology for the assisted delivery of intensive voice treatment (LSVT®LOUD) for Parkinson disease. *American Journal of Speech-Language Pathology*, 21(4):354-67, November, 2012. Recipient of the 2012 ASHA Editor's Award.
18. Heather Underwood S. Revi Sterling and John K. Bennett. Improving maternal labor monitoring in Kenya using digital pen technology: A user evaluation. In *Proceedings of the 2nd IEEE Global Humanitarian Technology Conference (GHTC 2012)*, Seattle, Washington, October 2012.
19. S. Revi Sterling and John K. Bennett. Three bad assumptions: Why technologies for social impact fail. *International Journal of Computational Engineering Research*. 2(4), 2012.
20. L. Dodson, S. R. Sterling, and J. K. Bennett. Considering failure: Eight years of ITID research. In *Proceedings of the Fifth IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD2012)*, Atlanta, Georgia, March 2012.
21. S. Revi Sterling and John K. Bennett. Crossing the real chasm in humanitarian technology work. *Proceedings of the 2011 IEEE Global Humanitarian Technology Conference (GHTC 2011)*. Seattle, USA. October 2011.
22. John K. Bennett and S. Revi Sterling. Computer science is not enough. *TripleC (Cognition, Communication, Co-Operation): Open Access Journal for a Global Sustainable Information Society*. 9(2):624-631, July 2011.
23. S. Revi Sterling, John W. O'Brien and John K. Bennett. Advancement through interactive radio. *Communication for Social Change*. 11(2), 2009.
24. Brian Shucker, Todd Murphey, and John K. Bennett. Convergence preserving switching for topology dependent decentralized systems. *IEEE Transactions on Robotics*. 24(6), 1405-1415, Dec. 2008.
25. S. Revi Sterling and John K. Bennett. User centric design for innovative women. *Information for Development*, 3(2), 49-51, July, 2008.
26. S. Revi Sterling, John W. O'Brien and John K. Bennett. Advancement through interactive radio. In *Proceedings of the 2nd IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD 2007)*. Bangalore, India. December 2007.
27. John Giacomoni, John K. Bennett, Antonio Carzaniga, Douglas C. Sicker, Manish Vachharajani, Alexander L. Wolf. Frame shared memory: Line-rate networking on commodity hardware. In *Proceedings of ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2007)*. Coronado Springs Resort, Orlando, Florida, USA. December 2007
28. S. Revi Sterling and John K. Bennett. Hear My Voice: A system to capture personal ethnographies from African mwethyas. In *Proceedings of the Third International Ethnographic Praxis in Industry Conference (EPIC 2007)*. Keystone, Colorado, USA. October 2007.
29. S. Revi Sterling and John K. Bennett. Broadcasting women: Participatory community radio as a space of empowerment. In *Proceedings of the Royal Geographic Society Annual International Conference (RGS 2007)*. London, Great Britain. August 2007.
30. S. Revi Sterling and John K. Bennett. Out of the margins: Lessons from Africa on including women in community radio. In *Proceedings of Community Radio India 2007 Conference (eIndia 2007)*. New Delhi, India. July 2007.
31. Brian Shucker, Todd Murphey, and John K. Bennett. Switching rules for decentralized control with simple control laws. In *Proceedings of the 2007 American Control Conference (ACS 2007)*. New York City, USA, July 2007.
32. Brian Shucker, Todd Murphey, and John K. Bennett. Testbed implementation of wireless distributed control. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007)*. Rome, Italy. April 2007.

33. S. Revi Sterling and J.K. Bennett. Broadcasting women, broadcasting resistance. In *Proceedings of the Society for Social Studies of Science* (4S) Vancouver, BC, November 2006.
34. Daniel Crawl, Joseph Dunn, Avneesh Bhatnagar, Evan Speight, and John Bennett. Using location dependence to manage mobile data. In *Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services* (MOBIQUITOUS 2006), San Jose, California. July, 2006.
35. S.R. Sterling and J.K. Bennett. Women's identity production through participatory community radio. In *Proceedings of the International Conference on Communication for Development and Social Change*. July, 2006.
36. Brian Shucker, Todd Murphey, and John K. Bennett. Switching control without nearest neighbor rules. In *Proceedings of the 2006 American Control Conference*. 2006.
37. Brian Shucker and John K. Bennett. Cooperative control using occasional non-local interactions. In *Proceedings of 2006 IEEE International Conference on Robotics and Automation*. June, 2006.
38. Graham Schelle, Daniel Fay, Dirk Grunwald, Dan Connors and John Bennett. An evolving curriculum to match the evolution of reconfigurable computing platforms. In *Proceedings of WRCE2006 (IEEE Computer Society Workshop on Reconfigurable Computing Education)*. March, 2006.
39. Brian Shucker and John K. Bennett. Target tracking with distributed robotic macrosensors. In *Proceedings of MilCom 2005 (IEEE/AFCEA Military Communications Conference)*, Oct. 2005, Atlantic City NJ.
40. J. Dunn, M. Neufeld, A. Sheth, D. Grunwald and J. Bennett. A practical cross-layer mechanism for fairness in 802.11 networks. *ACM/Kluwer Mobile Networks and Applications Journal*. December, 2005.
41. T. Mayes and J.K. Bennett. A survey of current ABET best practices. *Proceedings of the 2005 ASEE Annual Conference*. (ASEE 2005). June 2005. Portland, Oregon.
42. J. Dunn, M. Neufeld, A. Sheth, D. Grunwald and J. Bennett. A practical cross-layer mechanism for fairness in 802.11 networks. *Proceedings of the First Annual International Conference on Broadband Networks* (BroadNets 2004). October 2004. San Jose, California.
43. B. Shucker and J. Bennett. Scalable control of distributed robotic macrosensors. *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems* (DARS'04). June 2004. Toulouse, France
44. H. Abdel-Shafi, W.E. Speight and J.K. Bennett. Raptor: integrating checkpoints and thread migration for cluster management. In *Proceedings of the 22nd Symposium on Reliable Distributed Computing* (SRDS2003), 2003.
45. A. Bhatnagar, E. Speight, D. Crawl, J. Dunn, and J. K. Bennett. Application management techniques for the Bifrost system. In *Proceedings of the Fifth IEEE Workshop on Mobile Computing Systems & Applications*, October, 2003.
46. W.E. Speight, H. Abdel-Shafi, and J.K. Bennett. WSDLite: A lightweight alternative to Windows Sockets Direct Path. In *Proceedings of the 2000 USENIX Windows Systems Research Symposium*, August, 2000.
47. H. Abdel-Shafi, W.E. Speight and J.K. Bennett. Efficient user-level thread migration and checkpointing on Windows NT clusters. In *Proceedings of the 1999 USENIX Windows/NT Research Symposium*, 1-10, July, 1999.
48. W.E. Speight, H. Abdel-Shafi, and J.K. Bennett. Realizing the performance potential of the Virtual Interface Architecture. In *Proceedings of the 13th ACM International Conference on Supercomputing* (ICS), June 1999.
49. D.L. Hamilton, I.D. Walker, and J.K. Bennett. Parallel robot control using speculative computation. In *Journal of Robotics and Automation*, 13(4), 101-112, Dec, 1998.
50. W.E. Speight, H. Abdel-Shafi, and J.K. Bennett. An integrated shared-memory/message passing API for cluster-based multicomputing. In *Proceedings of the Second International Conference on Parallel and Distributed Computing and Networks*, December, 1998.

51. J.R. Nelson, T.A. Roming, J.K. Bennett. A whole glove method for evaluation of surgical gloves as barrier to virus. *Journal of Allergies and Clinical Immunology*, 1998.
52. W.E. Speight and J.K. Bennett. Using multicast and multithreading to reduce communication in software DSM systems. In *Proceedings of the Fourth Symposium on High Performance Architecture (HPCA)*, 312-323, February, 1998.
53. W.E. Speight and J.K. Bennett. Brazos: A third generation DSM system. In *Proceedings of the 1997 USENIX Windows/NT Workshop*, 95-106, August, 1997.
54. D.L. Hamilton, I.D. Walker, and J.K. Bennett. Parallel robot control using speculative computation. In *Journal of Robotics Systems*, 1997.
55. J.K. Bennett. The clinical significance of hydration in natural rubber latex gloves. In *Surgical Services Management*, 3(2), 29-33, Feb, 1997.
56. D.L. Hamilton, I.D. Walker, and J.K. Bennett. Fault tolerance versus performance metrics for robot manipulators. In *Reliability Engineering and System Safety*, 53(1996), 309-318, 1996.
57. J.K. Bennett, K.E. Fletcher, and W.E. Speight. The performance value of shared network caches in clustered multiprocessor workstations. In *Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS-16)*, May 1996.
58. Yanyang Xiao and J.K. Bennett. Memory organization in multi-channel optical networks: NUMA and COMA revisited. In *Proceedings of the 10th ACM International Conference on Supercomputing (ICS)*, May 1996.
59. D.L. Hamilton, I.D. Walker, and J.K. Bennett. Fault tolerance versus performance metrics for robot manipulators. In *Proceedings of the 1996 International Conference on Robotics and Automation*, 3073-3080, Minneapolis, MN, 1996.
60. D.L. Hamilton, I.D. Walker, and J.K. Bennett. Parallel robot control using speculative computation. In *Proceedings of the 1996 International Conference on Robotics and Automation*, 3420-3427, Minneapolis, MN, 1996.
61. J.B. Carter, J.K. Bennett, and W. Zwaenepoel. Techniques for reducing consistency-related communications in distributed shared memory systems. *ACM Transactions on Computers*, 13(3), 205-243, Aug. 1995.
62. J.K. Bennett, J.B. Carter, and W. Zwaenepoel. Adaptive software cache management for distributed shared memory architectures. In *The Cache Coherence Problem in Shared Memory Multiprocessors: Software Solutions*, Igor Tartalja and Veljko Milutinovic, editors, IEEE Computer Society Press, 1995.
63. J.K. Bennett. The feasibility of using electrical means for monitoring barrier integrity in natural rubber latex gloves. In *Journal of Infection Control and Hospital Epidemiology*, 1995.
64. R. Mukherjee and J.K. Bennett. Operating system design principles for scalable shared memory multiprocessors. In *Proceedings of the Symposium on Parallel and Distributed Computing Systems (PDCS-95)*, 1995.
65. R. Mukherjee, J.K. Bennett, and J.A. Greenwood. The effects of architecture on the performance of latency hiding via rapid context switching. In *Proceedings of the 7th IASTED International Conference on Parallel and Distributed Computing and Systems*, 1995.
66. J.K. Bennett. The feasibility of using electrical means for monitoring barrier integrity in natural rubber latex gloves. In *Proceedings of the American College of Surgeons / Center for Disease Control Conference on Bloodborne Pathogens*, February, 1994.
67. D.L. Hamilton, M.L. Visinsky, J.K. Bennett, J.R. Cavallaro, and I.D. Walker. Fault tolerant algorithms and architectures for robotics. In *Proceedings of the 1994 Mediterranean Electrotechnical Conference*, 1034-1036, Antalya, Turkey, April 1994.
68. J.K. Bennett. Hydration and conductivity in natural rubber latex gloves. *Source to Surgery* 1(3), 1-4, October, 1993.
69. J.K. Bennett. Investigating the physical properties of latex gloves. In *Touch* 2(2), 1-2, May 1993.

70. J.K. Bennett, S. Dwarkadas, J.A. Greenwood, and Evan Speight. Willow: A scalable shared memory multiprocessor. In *Proceedings of SuperComputing 92*, IEEE Computer Society Press, pp. 336-345, Nov. 1992.
71. D.L. Hamilton, J.K. Bennett, and I.D. Walker. Simulation of a reliable robot control architecture. In *Proceedings of the 1992 International Simulation Technology Conference*, 321-327, Nov. 1992.
72. D.L. Hamilton, J.K. Bennett, and I.D. Walker. Parallel fault-tolerant robot control. In *Proceedings of the 1992 SPIE Conference on Cooperative Intelligent Robotics in Space III*, pp. 251-261, Nov. 1992.
73. J.K. Bennett, J.B. Carter, A.L. Cox, E.N. Elnozahy, D.B. Johnson, P. Keleher and W. Zwaenepoel. Distributed shared memory: Experience with Munin. In *Proceedings of the Fifth European ACM SIGOPS Workshop*, July, 1992.
74. J.B. Carter, J.K. Bennett, and W. Zwaenepoel. Implementation and performance of Munin. In *Proceedings of the 13th Symposium on Operating System Principles*, pp. 152-164, Oct. 1991.
75. J.K. Bennett, J.B. Carter, and W. Zwaenepoel. Toward large scale shared memory multi-processing. In *Scalable Shared Memory Multiprocessors*, M. Dubois and Shreekanth Thakkar, editors, Kluwer Academic Publishers, Nov. 1991.
76. J.K. Bennett, J.B. Carter, and W. Zwaenepoel. Munin: Distributed shared memory using multi-protocol release consistency, In *Operating Systems of the 90s and Beyond*, A.I. Karshner and J. Nehmer, editors, Lecture Notes in Computer Science, Springer-Verlag LNCS 563, pp. 56-60, 1991.
77. J.K. Bennett, J.B. Carter, and W. Zwaenepoel. Adaptive software cache management for distributed shared memory architectures. In *Proceedings of the 17th International Symposium on Computer Architecture*, 125-134, May 1990.
78. J.K. Bennett, J.B. Carter, and W. Zwaenepoel. Munin: Distributed shared memory based on type-specific memory coherence. In *Proceedings of the Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pp. 168-176, Mar. 1990.
79. J.K. Bennett. Experience with Distributed Smalltalk. *Software Practice and Experience*, 20(2), 157-180, Feb. 1990.
80. R. Mukherjee and J.K. Bennett. Simulation of parallel programs on a shared memory multiprocessor. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS23)*, pp. 242-251, Dec. 1989.
81. J.K. Bennett. The design and implementation of Distributed Smalltalk. In *Proceedings of the Second Annual Conference on Object-Oriented Programming Systems, Languages, and Applications*, 318-330, Orlando, Florida, Oct. 1987.
82. J.K. Bennett. Raster operations: integrating text and graphics in high performance systems. *Byte*, 10(12): 1187-1192, Nov. 1985.

Appendix B.

Cases in Which I Have Testified at Trial or by Deposition as an Expert:

Haden and Company v. AMS (1990)

ResTech v. Joseph Hawkins (1993)

Siege Industries, Inc. v. Clark Manufacturing (1996)

Caldera, Inc. v. Microsoft Corporation (1997)

Bristol Technology, Inc. v. Microsoft Corporation (1999)

Compaq Computer Corporation v. eMachines Inc. (2000)

Compaq Computer Corporation v. Unova, Inc. et al. (2001)

State of New York ex. rel. v. Microsoft Corporation (2002)

Microsoft California consumer cases (2002)

Microsoft MDL consumer cases (2002)

Friedman v. Microsoft (2003)

Gordon v. Microsoft (2003)

New River Holding Limited Partnership v. Precision Response Corporation, et al. (2004)

SuperSpeed Software, Inc. v. Oracle Corporation (2005)

Comes et al. v. Microsoft Corporation (2006)

Paltalk Holdings, Inc. v. Microsoft Corporation (2008)

SuperSpeed Software, Inc. v. IBM Corporation (2008)

Novell Inc. v. Microsoft Corporation (2009)

GO Computer, Inc. and S. Jerrold Kaplan v. Microsoft Corporation (2009)

NXP Semiconductors USA, Inc. v. Exatel Visual Systems, Ltd. (2011)

SuperSpeed Software, Inc. v. Google Inc. (2013)

Pro-Sys Consultants Ltd. and Neil Godfrey v. Microsoft Corporation and Microsoft Canada Co. / Microsoft Canada CIE (2017)

Appendix B.

In Re Volkswagen Clean Diesel Litigation: TCAA Enforcement Case;
Cause No. D-1-GN-16-000370 (2022)

International Business Machines Corporation (IBM) v. Zynga Inc., C.A. No.
22-590-GBW (2024)

I have also provided consulting and expert opinions not involving testimony in other matters. This other work is in general governed by protective orders or non-disclosure agreements.